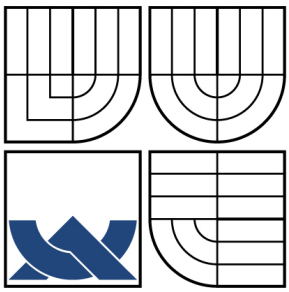
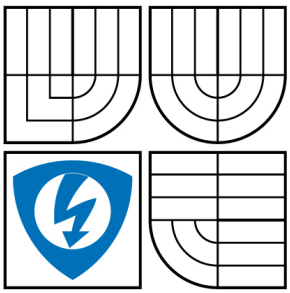


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

AUTOMATICKÉ LADĚNÍ VAH PRAVIDLOVÝCH BÁZÍ ZNALOSTÍ

AUTOMATED WEIGHT TUNING FOR RULE-BASED KNOWLEDGE BASES

DIZERTAČNÍ PRÁCE

DOCTORAL THESIS

AUTOR PRÁCE — Ing. JAN VALENTA

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. VÁCLAV JIRSÍK, CSc.

SUPERVISOR

BRNO 2009

Poděkování

Rád bych poděkoval vedoucímu disertační práce doc. Ing. Václavu Jirsíkovi za metodické vedení a cenné rady při výzkumu a zpracování mé disertační práce.

Také bych rád poděkoval Ing. Tomáši Panáčkovi za pomoc při zpracování a implementaci diferenciálního evolučního algoritmu. Zvláštní poděkování patří také MUDr. Vendule Jandlové za odborné rady, materiály a pomoc při tvorbě báze znalostí pro diagnostiku poruch srdečního rytmu.

Abstrakt

Předložená disertační práce představuje nové možnosti automatizované tvorby a ladění bází znalostí informačních a expertních systémů. Práce je rozdělena na dvě navazující části. První část se věnuje existujícímu expertnímu systému NPS32 vyvinutému na Fakultě elektrotechniky a komunikačních technologií, Vysokého učení technického v Brně. Matematický aparát tohoto systému je založen na vyjádření neurčitosti pravidel dvěma hodnotami, čímž rozšiřuje informační schopnosti báze o hodnoty nepřítomnosti informace a sporu v bázi znalostí. Program byl doplněn učícím algoritmem nastavujícím váhy pravidel v bázi znalostí pomocí diferenciálního evolučního algoritmu za pomoci vzorů získaných od experta. Vzory představují reakci systému na data vložená uživatelem. Uvedený učící algoritmus je omezen pouze na jednovrstvé báze znalostí. V práci je uveden formální důkaz nemožnosti použití matematického aparátu expertního systému NPS32 pro učení vícevrstvých bází znalostí gradientními metodami. Druhá část práce se věnuje učícímu algoritmu pro realizaci vícevrstvé pravidlové báze znalostí. Báze znalostí je založena na specifickém modelu pravidla ohodnoceného neurčitostí vyjadřující míru informačního přínosu pravidla. Jako algoritmus učení nastavující váhy jednotlivých pravidel ve struktuře báze je použit modifikovaný algoritmus back-propagation přepracovaný pro použití s danou strukturou báze a modelem pravidla. Pro účely testování a ověření učícího algoritmu ladění báze znalostí byl vytvořen expertní systém RESLA v jazyce C#. V tomto expertním systému byla vytvořena báze znalostí z oblasti medicíny k ověření učících schopností algoritmu pro složité báze znalostí. Báze znalostí tvoří diagnostiku poruch srdečního rytmu na základě parametrů získaných z průběhu EKG (elektrokardiogramu). Za účelem srovnání s již existujícími bázemi znalostí tvořenými expertem a znalostním inženýrem byl program srovnáván s profesionálně odladěnou bází znalostí z oblasti zemědělství. Bázi zde tvoří systém na podporu rozhodování při výběru vhodné odrůdy pšenice ozimé pro pěstování v různých prostředích. Tato disertační práce řeší zásadní zrychlení tvorby bází znalostí při zachování všech výhod, která plynou z použití pravidel. Oproti současným řešením založeným na neuronových sítích je stávající algoritmus pro ladění bází znalostí rychlejší a jednodušší, neboť nevyžaduje extrakci pravidel z jiného typu báze znalostí.

Abstract

This dissertation thesis introduces new methods of automated knowledge-base creation and tuning in information and expert systems. The thesis is divided in the two following parts. The first part is focused on the legacy expert system NPS32 developed at the Faculty of Electrical Engineering and Communication, Brno University of Technology. The mathematical base of the system is expression of the rule uncertainty using two values. Thus, it extends information capability of the knowledge-base by values of the absence of the information and conflict in the knowledge-base. The expert system has been supplemented by a learning algorithm. The learning algorithm sets weights of the rules in the knowledge-base using differential evolution algorithm. It uses patterns acquired from an expert. The learning algorithm is only one-layer knowledge-bases limited. The thesis shows a formal proof that the mathematical base of the NPS32 expert system can not be used for gradient tuning of the weights in the multilayer knowledge-bases. The second part is focused on multilayer knowledge-base learning algorithm. The knowledge-base is based on a specific model of the rule with uncertainty factors. Uncertainty factors of the rule represents information impact ratio. Using a learning algorithm adjusting weights of every single rule in the knowledge base structure, the modified back-propagation algorithm is used. The back-propagation algorithm is modified for the given knowledge-base structure and rule model. For the purpose of testing and verifying the learning algorithm for knowledge-base tuning, the expert system RESLA has been developed in C#. With this expert system, the knowledge-base from medicine field, was created. The aim of this knowledge-base is verify learning ability for complex knowledge-bases. The knowledge-base represents heart malfunction diagnostic base on the acquired ECG (electrocardiogram) parameters. For the purpose of the comparison with already existing knowledge-basis, created by the expert and knowledge engineer, the expert system was compared with professionally designed knowledge-base from the field of agriculture. The knowledge-base represents system for suitable cultivar of winter wheat planting decision support. The presented algorithms speed up knowledge-base creation while keeping all advantages, which arise from using rules. Contrary to the existing solution based on neural network, the presented algorithms for knowledge-base weights tuning are faster and more simple, because it does not need rule extraction from another type of the knowledge representation.

OBSAH

Seznam použitých zkratek

ANN	umělá neuronová síť (artificial neural network)
BP	metoda učení zpětným šířením chyby (back-propagation)
CF	faktor (ne)jistoty (certain factor)
EKG	elektrokardiogram
ES	expertní systém
GUI	grafické uživatelské rozhraní (graphical user interface)
UI	umělá inteligence
UML	grafický jazyk pro modelování, specifikaci a dokumentaci programových systémů (unified modeling language)

Seznam obrázků

Seznam tabulek

1 Úvod

Počátky expertních systémů, jako vědního oboru umělé inteligence, jsou datovány do poloviny sedmdesátých let dvacátého století. V tomto období dochází ke změně ve vývoji umělé inteligence od hledání univerzálního algoritmu k otázce reprezentace a zpracování znalostí.

V následujících letech vznikaly rozličné reprezentace znalostí, z nichž mezi nejúspěšnější a dnes nejpoužívanější patří výroková logika a neuronové sítě (ANN). Těmto dvěma způsobům reprezentace znalostí bylo věnováno obrovské množství času při výzkumu a vznikly tak mnohá zdokonalení ve formě zpracování neurčitosti, učících algoritmů apod., které jim zajistilo úspěch i v současné době. Přesto však oba typy reprezentace mají své nedostatky, které částečně omezují jejich použití v různých oborech a prostředích.

Velkou výhodou uchování znalostí ve formě báze pravidel, které se používají v expertních a informačních systémech, je srozumitelnost báze znalostí, jednoduchost, modularita a možnost použití vysvětlovacího mechanizmu poskytujícího uživateli informace jak a proč byly odvozeny výstupy, nebo závěry při konzultaci. Časová a finanční náročnost je naopak významným limitujícím faktorem při tvorbě a ladění pravidlových bází znalostí.

Neuronové sítě vynikají efektivními učicími algoritmy, při zpracování velkého množství dat a při nedostatečné strukturovanosti problému. Na druhé straně jsou takto uložené znalosti velmi těžko srozumitelné a je tedy i těžko ověřitelné, zda báze znalostí korektně a úplně popisuje oblast řešené problematiky.

Nabízí se otázka, zda nelze najít algoritmus nebo reprezentaci znalostí, která by dokázala využít výhod obou těchto typů reprezentací znalostí.

Předložená disertační práce se zaměřuje na postupy tvorby a ladění báze znalostí, které dokáží využít cenných vlastností obou zmíněných reprezentací znalostí, tedy pravidel a neuronových sítí. Dosavadní vývoj algoritmů a metod pro kombinování neuronových sítí a pravidel je popsán v kapitole 2.

Na základě poznatků získaných rozborem stávajících metod tvorby bází znalostí kombinováním pravidel a neuronových sítí byly v třetí kapitole stanoveny cíle disertace.

Práce představuje dva algoritmy a dvě odlišné reprezentace znalostí založené na množině pravidel s ohodnocením každého pravidla neurčitostí.

Ve čtvrté kapitole je popsán algoritmus, který je navržen jako optimalizátor pro báze znalostí již existujícího expertního systému NPS32 vyvinutého na Fakultě elektrotechniky a komunikačních technologií, Vysokého učení technického v Brně. Optimalizačním prvkem je zde diferenciální evoluční algoritmus. V této kapitole je popsán vlastní matematický aparát expertního systému NPS32, matematická reprezentace a struktura báze znalostí, metoda učení a optimalizace vah pravidel báze znalostí na základě stanovené kriteriální funkce. Optimalizace vah pravidel byla ověřena odladěním existující báze znalostí. V závěru kapitoly je proveden formální důkaz, že matematický aparát expertního systému NPS32 nelze použít pro gradientní optimalizaci

u vícevrstvých bází znalostí. Z tohoto důvodu byl vytvořen druhý algoritmus, který umožňuje ladění vah pravidel v libovolně strukturované dopředné bázi znalostí.

V páté kapitole je popsán druhý algoritmus, který využívá metodu přímého ladění vah pravidel v bázi znalostí podobným způsobem jako algoritmus back-propagation u neuronových sítí. Získáme tak nejen výhodu rychlého a jednoduchého učení, při zachování všech vlastností pravidlové sítě, ale i možnost učení této sítě z dat, což dříve nebylo možné. Otevře se tak také možnost plně automatické tvorby pravidlových sítí včetně ohodnocení pravidel neurčitostí. Kapitola obsahuje matematický popis báze znalostí a popis její struktury. Je zde odvozen algoritmus optimalizace vah pravidel báze znalostí na základě stanovené kriteriální funkce. Optimalizační algoritmus byl experimentálně ověřen odladěním vah pravidel ve třech reálných bázích znalostí popsaných v kapitole 6.

Vlastní programové zpracování algoritmů a expertních systémů je popsáno v sedmé kapitole. První část kapitoly je věnována expertnímu systému NPS32, který implementuje evoluční optimalizační algoritmus, druhá část se věnuje expertnímu systému RESLA, do kterého byl implementován algoritmus přímého ladění vah pravidel v bázi znalostí. V kapitole je popsáno programové zpracování expertních systémů a algoritmů, datové struktury a důležité funkce jednotlivých modulů obou expertních systémů. Pro expertní systém RESLA je uveden vývojový cyklus báze znalostí. Jednotlivé kapitoly popisují práci s příslušnými moduly expertních systémů.

2 Využití kombinace pravidel a neuronových sítí pro tvorbu bází znalostí

V této kapitole jsou shrnuty dosavadní metody tvorby pravidlových bází znalostí použitím extrakce pravidel z naučené neuronové sítě. Jsou zde také uvedeny metody tzv. čištění pravidel, které umožňují zpřesnění báze pravidel modelované domény a metody inicializace neuronové sítě známými pravidly, které urychlují proces učení. V závěru kapitoly jsou shrnuty nejdůležitější vlastnosti metod z této kapitoly.

Tvorba báze znalostí je všeobecně považována za nejnáročnější část celého vývojového cyklu expertního systému. Zahrnuje analýzu problematiky, zda je vůbec možné bázi vytvořit, návrh reprezentace znalostí pro příslušnou doménu, získání znalostí od expertů nebo z dat, implementaci těchto znalostí ve formě příslušného matematického nebo logického zápisu a odladění a testování funkčnosti a generalizace báze. Omezíme-li se na nejpoužívanější pravidlové báze znalostí, je právě tou nejproblematičtější částí vývoje ladění jednotlivých vah pravidel báze. Ladění báze vah pravidel báze znalostí představuje iterativní proces, kdy jsou váhy pravidel nastavovány tak, aby báze znalostí co nejpřesněji popisovala modelovanou doménu. V podání znalostního inženýra může tento proces pro středně velké báze znalostí trvat i několik měsíců. U velkých bází můžeme hovořit i o horizontu let. Doba vývoje se ovšem zásadně liší podle zvolené reprezentace znalostí. S příchodem neuronových sítí se výrazně rozšířily možnosti tvorby bází znalostí z dat pomocí vzorů. K úspěšnosti ANN přispívají zejména tyto čtyři charakteristické rysy [24]:

- způsob, kterým neuronové sítě získávají znalosti o dané doméně prostřednictvím trénovací fáze,
- kompaktnost formy, ve které jsou znalosti uloženy, ačkoliv jsou ve zcela numerické formě,
- robustnost řešení získaného pomocí ANN za přítomnosti šumu,
- schopnost generalizace trénovacích vzorů.

Hovoříme zde o tzv. neuroexpertních systémech, které nejčastěji využívají jako bázi znalostí vícevrstvou dopřednou perceptronovou síť. Problémem této reprezentace znalostí je její neprůhledná struktura a uzavřenost znalostí reprezentovaných množinou reálných vah, prahů a přenosových funkcí jednotlivých neuronů.

Obrovskou výhodou pravidlových bází jsou poměrně velké možnosti vysvětlování postupů a odvození hypotéz, které umožňují nasazení těchto systémů i v oblastech tzv. bezpečných a kritických, jako jsou energetika, medicína, letecký průmysl a další. Pravidla také umožňují velmi dobrou verifikovatelnost, a to i na úrovni vnitřních stavů báze. Neuronové sítě tuto vlastnost postrádají. Zkušenosti ukazují, že právě špatné vysvětlovací a ověřovací schopnosti jsou příčinou nízkého průmyslového využití neuronových sítí jako prostředku pro tvorbu bází znalostí [24].

Velmi dobré schopnosti učení ANN a možnosti získávání nových, dříve nepoznaných, znalostí z dat později způsobily zájem vědců o neuronové sítě jakožto

reprezentaci znalostí. Rozvoj bází znalostí reprezentovaných neuronovými sítěmi přispěl ke vzniku mnoha algoritmů umožňujících, více či méně, srozumitelně vysvětlit postup, jakým byly odvozovány hypotézy a závěry expertního systému. Tyto algoritmy také zahrnují velkou skupinu, ve které je umožněno nejen vysvětlování odvozovacích postupů v neuronové síti, ale také využívají metody pro tvorbu pravidlových bází znalostí z natrénované neuronové sítě. Další skupina algoritmů slouží k tzv. čištění pravidel za pomoci neuronové sítě. Jde o proces, kdy jsou nekompletní a nekonzistentní pravidla transformována do neuronové sítě. Tato síť je následně natrénována na učicí vzory na požadovanou přesnost a pravidla jsou zpětně extrahována. Je dokázáno [39], že výsledná pravidla mohou danou doménu lépe reprezentovat, než naučená neuronová síť.

2.1 Srovnání hlavních rysů symbolických metod a metod založených na neuronových sítích

V této kapitole jsou uvedeny principy činnosti a vlastnosti pravidlových systémů, systémů založených na neuronových sítích a systémů kombinujících oba tyto přístupy.

2.1.1 Pravidlové expertní systémy

Pravidlové expertní systémy jsou tvořeny bází znalostí (pravidla strukturovaná nejčastěji ve formě rozhodovacího stromu), bází dat (fakta k danému případu) a inferenčního mechanizmu. Pravidla rozhodovacího stromu tvoří nejčastěji implikace mezi předpokladem a cílovou hypotézou, případně mezivýsledkem. Z dat dosazených do antecedentů pravidel odvozuje inferenční mechanizmus nová fakta. Existují dva způsoby řízení inference. Dopředné řetězení, kdy jsou na počátku známa všechna fakta k případu a jejich aplikací se inferenční mechanizmus snaží dosáhnout cíle. Druhým způsobem je zpětné řetězení, kdy inferenční mechanizmus vychází ze známého cíle a rekurzivně vybírá pravidla, které tento cíl podporují, dokud nejsou určena všechna potřebná fakta [19], [27].

Důležitou schopností inferenčního mechanismu je zpracování neurčitosti. Neurčitost může být reprezentována a zpracovávána např. pomocí pravděpodobnostního (Bayesovského) přístupu, faktorů (ne)jistoty, Dempster-Shaferovy teorie, nebo fuzzy logiky. Následující odstavce převážně popisují práci pomocí faktorů jistoty (CF - certain faktor). Faktory jistoty nabývají hodnot z intervalu [-1,1]. Pravidla pak mají tvar: $E \rightarrow H(CF)$ [27], kde E (evidence) tvoří množinu antecedentů a H (hypothesis) je konsekvent.

Skládání neurčitosti (konjunkce a disjunkce) pomocí faktorů jistoty je podle následujících vztahů [5]:

$$\textit{if } A \textit{ and } B \textit{ then } C(CF): \quad CF(C) = \min(CF(A)\cdot CF, CF(B)\cdot CF), \tag{2.1}$$

$$\textit{if } A \textit{ or } B \textit{ then } C(CF): \quad CF(C) = \max(CF(A)\cdot CF, CF(B)\cdot CF). \tag{2.2}$$

2.1.2 Neuronové sítě

Neuronové sítě jsou tvořeny vzájemně propojenými neurony s vlastní přenosovou a prahovací funkcí. Neurony jsou uspořádány do vrstev, kde rozlišujeme vstupní vrstvu, skryté vrstvy a výstupní vrstvu neuronů. Jednotlivé neurony jsou navzájem spojeny hranami. Hrany představují jednotlivé váhy neuronů a jsou ohodnoceny váhovým koeficientem, určujícím míru uplatnění vstupní hodnoty neuronu na jeho výstup. Hrany a uzly tvoří dohromady topologii sítě, která se udává jako vektor počtu prvků (neuronů) v každé vrstvě sítě. ANN je paralelní výpočetní jednotka, která na základě vstupních hodnot po vrstvách přepočítává výstupy jednotlivých neuronů dokud nedojde k výstupní vrstvě [26].

V dále zmiňovaných algoritmech extrakce pravidel (kapitola 2.2 - 2.4) uvažuji, pokud není řečeno jinak, algoritmus učení neuronové sítě back-propagation (BP), který používá vzory (dvojce vektorů vstupních a výstupních hodnot) ke zjištění chybové funkce sítě. Tuto chybu zpětně šíří sítí a na jejím základě ovlivňuje příslušné váhy jednotlivých neuronů [16].

2.1.3 Hybridní systémy

Hybridní systémy využívají několik různých spolupracujících reprezentací znalostí k dosažení lepších parametrů pro zpracování dané domény. Velmi častou kombinací jsou ANN a pravidla. Způsob jejich kombinování je popsán v kapitolách 2.2 - 2.5. Tabulka 1 ukazuje vzájemné mapování těchto dvou reprezentací znalostí.

Pravidlové sítě	Neuronové sítě
hypotézy, cíle	výstupní neurony
vstupní uzly, antecedenty	vstupní neurony
mezilehlé uzly	skryté neurony
závislosti	vážená spojení neuronů

Tabulka 1: Mapování pravidlových a neuronových sítí [5]

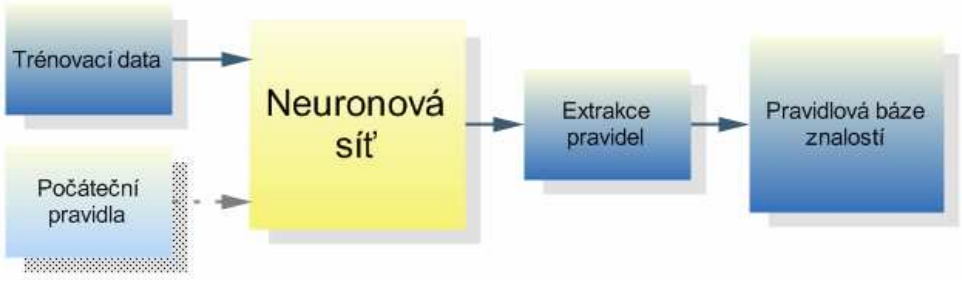
Tabulka 2 ukazuje komplementárnost vlastností NN a pravidel (symbolické reprezentace).

	Pravidlové sítě	Neuronové sítě
Reprezentace	strukturovaná	nestrukturovaná
Inference	přesná	přibližná
Generalizace	špatná	velmi dobrá
Vysvětlování	velmi dobré	špatné
Adaptace	střední	velmi dobrá

Tabulka 2: Srovnávání vlastností neuronových sítí a symbolické reprezentace [22]

2.2 Extrakce pravidel z naučené ANN

Reprezentace znalostí pravidly se díky své vynikající srozumitelnosti a jednoduchosti vyvinula do etalonu, se kterým jsou ostatní reprezentace znalostí srovnávány při ověřování konzistentnosti znalostní báze, případně pro srovnání vysvětlovací schopnosti. Vzniklo tak množství algoritmů, které transformují znalosti z naučené neuronové sítě na pravidla, aby rozšířili možnosti vysvětlování a ověřování konzistence této reprezentace znalostí. Jde tedy o úlohu převodu znalostí zakódovaných množinou reálných vah a prahů neuronů, případně i struktury sítě, na logicky vyjádřená pravidla. Tato úloha se nazývá extrakce pravidel z natrénované ANN (obrázek 1).



Obrázek 1: Extrakce pravidel z naučené ANN

Extrakci pravidel můžeme definovat takto:

Na základě naučené ANN a vzorů použitých k učení, produkuje extrakce pravidel stručný a přesný symbolický popis dané neuronové sítě [6].

Hlavními důvody pro extrakci pravidel z naučené ANN jsou [1]:

- získání vysvětlovací schopnosti,
- akvizice znalostí pro symbolické systémy,
- prozkoumávání dat a indukce vědeckých teorií,
- zvýšení schopnosti generalizace pro řešení s ANN,
- ověřování softwaru a ladění komponent založených na ANN.

V současné době existuje velké množství algoritmů transformujících data z naučené ANN na pravidla. Je proto dobré stanovit kategorie do kterých tyto jednotlivé algoritmy patří a představit pouze typické reprezentanty dané kategorie.

V práci „The True Will Come to Light: Directions and Challenges in Extracting the Knowledge Embedded Within Artificial Neural Networks“ autoři Alan B. Tickle a další [37] uvedli pět základních kategorií dělení algoritmů extrakce. Klasifikační kritéria byla stanovena takto [37], [1]:

- formát pravidel,
- kvalita pravidel,
- detailnost (v originále doslovně průhlednost),
- výpočetní náročnost,
- přenosnost.

Formát pravidel specifikuje tvar extrahovaných pravidel. Lze jej rozdělit na:

- propoziční logiku, tzv. Boolean pravidla,
- pravidla založená na fuzzy množinách a fuzzy logice.

Kvalita pravidel reprezentuje jejich vyjadřovací schopnost a přesnost. Autoři dále tyto kategorie dělí na:

- přesnost pravidel, tedy přesnost klasifikace příkladů, na které nebyla síť trénována,
- věrnost popisu ANN, tedy jak přesně odpovídá chování extrahovaných pravidel chování původní ANN,
- konzistentnost, určuje, zda pro různé trénovací množiny ANN vyhodnocuje extrahovaná pravidlová síť neznámé příklady stále stejně,
- srozumitelnost, vyjadřuje míru počtu pravidel v extrahované bázi znalostí a počtu antecedentů každého pravidla.

Detailnost určuje jak moc do hloubky extrahovaná pravidla popisují natrénovanou ANN. Zde můžeme algoritmy rozdělit na tři skupiny:

- dekompoziční, které sledují chování jednotlivých neuronů a toto chování popisují pravidly,
- pedagogické, které se dívají na ANN jako na černou skříňku a popisují chování ANN jako celku nezávisle na její struktuře a vnitřních funkcí neuronů,
- tzv. elektické, které kombinují oba předešlé případy.

Výpočetní náročnost určuje komplexnost jádra algoritmu pro extrakci pravidel.

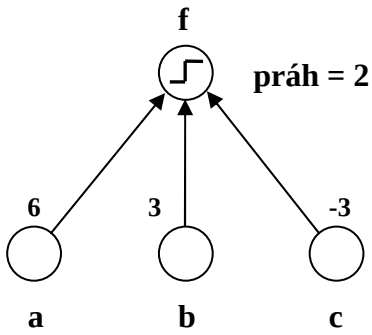
Přenosnost určuje parametr, zda daná extrakční technika může být použita pro více architektur ANN. Typickým představitelem jsou zde pedagogické algoritmy, které jsou na vnitřní struktuře ANN nezávislé.

Jako hlavní kritérium při dělení algoritmů extrakce pravidel bude použita detailnost. Algoritmy založené na extrakci fuzzy pravidel [28], [3], [13] jsou mimo rámec uvažované problematiky, neboť neodpovídají našim požadavkům na srozumitelnost, ačkoliv mohou poskytovat vyšší věrnost popisu. Znalosti ve fuzzy pravidlech jsou kódovány ve formě logicky spojených fuzzy množin s danou funkcí příslušnosti. Takto daná pravidla sama o sobě nemají o mnoho menší srozumitelnost než klasická boolean pravidla. Uvážíme-li ale proces odvozování znalostí v síti fuzzy pravidel, realizovaný vyhodnocováním, které je nejčastěji počítáno jako těžiště ploch pod funkcí příslušnosti aktivních pravidel (COG – Centre Of Gravity), stává se tento popis pro člověka jen velmi těžko srozumitelný. Omezíme se proto jen na logická pravidla.

2.3 Extrakce pravidel použitím dekompozičního přístupu

Kapitola popisuje extrakci pravidel metodami, které sledují chování jednotlivých neuronů a toto chování popisují pravidly. Metody dekomponují naučenou ANN na jednotlivé neurony a jejich analýzou vytvářejí pravidla.

Většina dekompozičních metod extrakce je založena na jednoduché neuronové síti (obrázek 2) s neuronem s prahovací funkcí [6].



Obrázek 2: Neuron s prahovací funkcí [6]

Metody předpokládají hodnoty vstupů a výstupů blízké jedničce, tedy maximálně aktivní, nebo nule, neaktivní. Z takovéto neuronové sítě můžeme jednoduše extrahovat prpoziční pravidla ve tvaru *if – then*. Aktivace neuronu na obrázku 2 je dána:

$$a_i = \begin{cases} 1 & když \sum_j w_{ij} a_j > \theta, \\ 0 & jinak, \end{cases} \tag{2.3}$$

kde a_j je hodnota vstupního neuronu j,

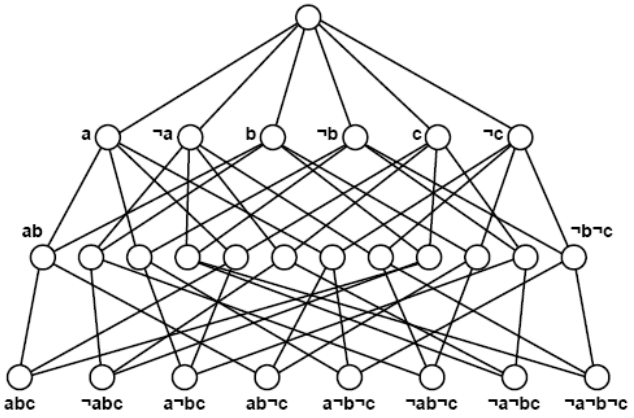
w_{ij} je váha z neuronu j k neuronu i,

θ_i je práh výstupního neuronu.

Extrahovaná pravidla (pro obrázek 2) jsou pak ve tvaru:

$$a \rightarrow f$$
$$b \wedge \neg c \rightarrow f$$

Obrázek 3 ukazuje prohledávaný stavový prostor pro neuron z obrázku 2.



Obrázek 3: Prohledávaný stavový prostor neuronu [6]

Každý uzel ve stavovém prostoru odpovídá jednomu potencionálnímu konjunktivnímu pravidlu. Vrchní uzly představují obecná pravidla, spodní uzly specifická pravidla.

S kombinací neuronových sítí a pravidel (za účelem využití výhod plynoucích z kombinace těchto dvou reprezentací znalostí) se poprvé setkáváme v roce 1989 v práci Li-Min Fu s názvem *Integration of Neural Heuristic into Knowledge-based Inference* [8]. Fu zde představuje algoritmus transformace pravidel na neuronovou síť, učení sítě a poukazuje na možnost zpětné extrakce pravidel. Tato práce je v současné době považována za překonanou, i když v oblasti zachování sémantiky sítě a jednoduchosti jistě překonána nebyla.

2.3.1 Algoritmus KT

Algoritmus KT [8] extrahuje pravidla z ANN se speciální strukturou a umožňuje inicializovat síť ze známých pravidel. ANN je učena kombinací algoritmů back-propagation a heuristiky hill-climbing.

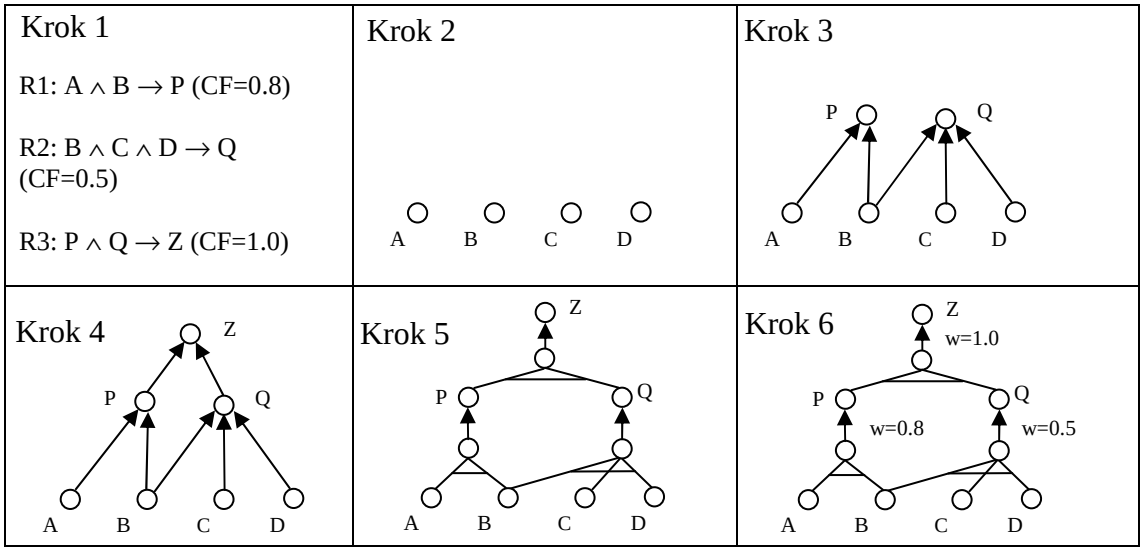
2.3.1.1 Sémantická tvorba neuronové sítě z pravidel

Tento algoritmus převádí pravidla (z pravidlové sítě) v konjunktivní formě do neuronové sítě tak, že vstupní uzly (dotazy, antecedenty pravidel) v pravidlové síti odpovídají vstupům ANN, odpovědi (výsledky, konsekventy pravidel) odpovídají výstupům ANN a vnitřní uzly skrytým neuronům ANN. Takto navržená transformace by sama o sobě nedodržovala sémantiku pravidel, na které je tato metoda založena, proto algoritmus vkládá mezi výstupy pravidla (daná vrstva sítě, vstup do následující vrstvy ANN) a vazby tvořící konjunkci antecedentů další neuron, který tvoří jakýsi most mezi podmínkovou a důsledkovou částí pravidel.

Postup převodu je popsán následujícími kroky [5]:

1. Pravidla jsou přepsána do konjunktivní formy.
2. Vstupní (datové) proměnné jsou mapovány na vstupní prvky ANN.
3. Vnitřní uzly (mezivýsledky) jsou mapovány na skryté prvky ANN.
4. Cílové proměnné jsou mapovány na výstupy ANN.
5. Jsou přidány spojovací neurony mezi podmínkovou a důsledkovou část pravidla.
6. Faktory (ne)určitosti jsou mapovány na příslušné spojení mezi spojovací neurony a důsledkovou část. Spojení mezi konjunktivními a nekonjunktivními vrstvami jsou nastaveny na 1.

Popsaný postup ukazuje na příkladu obrázek 4 [5].



Obrázek 4: Šest kroků převodu pravidel na neuronovou síť [5]

Ve výsledné ANN hledáme vektor vah reprezentující faktory určitosti v síti pravidel. Spojovací neurony reprezentují funkci minima antecedentů ve smyslu funkce (2.1). Váhy antecedentů pravidel jsou stále nastaveny na 1, čímž se autor vyhýbá těžko matematicky řešitelnému hledání vah, neboť funkce AND není diferencovatelná. Tyto váhy se nikdy nemění. Síť je rozdělena na konjunktivní a nekonjunktivní vrstvy. Nekonjunktivní vrstvy, jež obsahují hledané vektory faktorů určitosti, jsou diferencovatelné a mohou být nastavovány algoritmem BP. Pro konjunktivní vrstvy tento algoritmus použít nelze, proto algoritmus využívá mechanizmus hill-climbing k nalezení nejlepší cesty šíření chybové funkce. Chyba vypočtená např. v uzlu „Z“ obrázku 4 může být způsobena uzly „P“ nebo „Q“. Metoda testuje, který z těchto uzlů přináší větší chybu testováním jednoho po druhém. Výběr daného uzlu se provádí nastavováním vah příslušného uzlu. Po srovnání uzlů se nastaví váha u toho uzlu, který přináší největší zlepšení [5].

2.3.1.2 Extrakce pravidel

Z naučené neuronové sítě Fu extrahuje produkční pravidla pomocí jednoduchého algoritmu KT [9]. Pro každý neuron sítě (vnitřní i výstupní) a všechny ohodnocení jeho vstupů je vytvořeno logické pravidlo, jehož ohodnocení je dáno následujícím schématem [1]:

$$if\ \ 0 \le výstup\ pravidla \le práh\,1 \Rightarrow ohodnocení = False$$

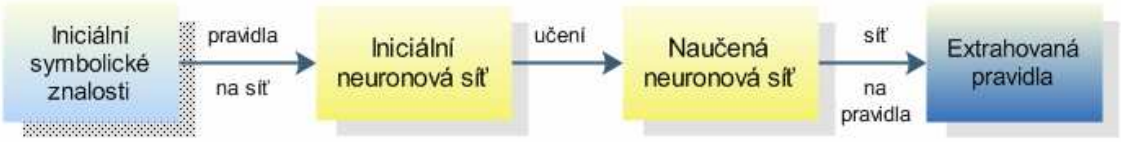
$$if\ \ práh\,2 \le výstup\ \ pravidla < 1 \Rightarrow ohodnocení = True$$

$$kde\ práh\,1 \le práh\,2.$$

2.3.2 Algoritmus KBANN

Další metodu využívající kombinaci neuronové sítě a pravidel popisují v roce 1994 autoři Towell a Shavlik v práci *KBANN* [40]. Metoda využívá pravidel k inicializaci neuronové sítě a následně je učena algoritmem back-propagation. Ve své disertační práci [38] se Towell věnuje i zpětné extrakci pravidel, která byla a je dále rozpracovávána mnohými dalšími autory.

KBANN (Knowledge Based Artificial Neural Network) je hybridní systém kombinující pravidla a neuronovou síť. Základem tohoto algoritmu je inicializace neuronové sítě pravidly, učení sítě algoritmem BP a případná extrakce pravidel zpět ze sítě [32]. Model tohoto systému je na obrázku 5.

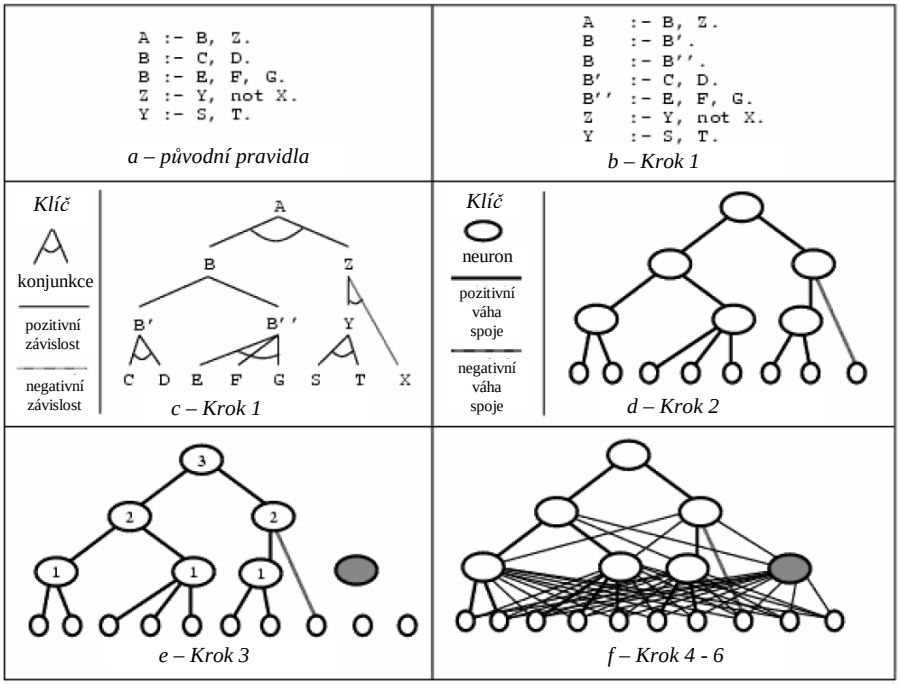


Obrázek 5: Model algoritmu KBANN [39]

Algoritmus převodu pravidel do NN je uveden v následujících šesti krocích [5], [4],[45]:

1. Pravidla transformujeme do Hornových klauzulí. Pravidla se stejným konsekventem $\{C \wedge D \rightarrow B, E \wedge F \wedge G \rightarrow B\}$ přepíšeme následujícím způsobem: $\{C \wedge D \rightarrow B', E \wedge F \wedge G \rightarrow B'', B' \vee B'' \rightarrow B\}$.
2. Přepíšeme pravidla do neuronové sítě. Negované atomy pomocí negativních vazeb, pozitivní atomy pomocí pozitivních vazeb, přičemž konjunkci konstruujeme tak, aby práh určoval nutnost splněných všech atomů. Váhy jsou tedy nastaveny tak, aby práh emuloval AND funkci. Towel a Shavlik empiricky našli, že dobrá iniciální váha je w=4 (případně w=-4 pro negaci). Práh je potom nastaven na $(P - 0.5) \cdot w$, kde P je počet kladných antecedentů. Pro disjunktivní vazbu je práh nastaven na $\frac{w}{2}$.
3. Každému neuronu přiřadíme číslo odpovídající jeho úrovni (definováno jako nejdelší cesta ke vstupnímu neuronu).
4. Přidáme vstupní neurony na základě odhadu experta.
5. Doplníme chybějící vazby a jejich váhy inicializujeme náhodnými čísly blízkými nule.
6. Takto vytvořenou síť můžeme klasickými algoritmy natrénovat na určité chování.

Popsaný algoritmus je uveden na jednoduchém příkladu:



Obrázek 6: Postup kroků algoritmu KBANN [45]

Z kroku dvě obrázku 6 je vidět, že systém aproximuje funkci AND. Algoritmus také nebere v úvahu neurčitost původních pravidel. Towell a Shavlik tvrdí, že vyjádření neurčitosti pravidel není nutné, neboť není zaručena jeho správnost [5]. Ztrácí se tak původní sémantika pravidel.

Autoři testovali KBANN oproti standardní ANN. Oba systémy testovali se stejnými daty různých velikostí, stejnými funkcemi a algoritmem učení BP. Pro malé množiny vzorů dává KBANN lepší výsledky, než standardní ANN. Se zvětšující se množinou vzorů tento rozdíl klesá. Ukazuje to, že doménové znalosti vložené do KBANN urychlují proces učení [5].

K extrakci pravidel z takto naučené ANN lze použít algoritmy Subset, nebo M of N.

2.3.3 Algoritmus Subset

Metoda extrakce pravidel z neuronové sítě vychází ze dvou předpokladů [39]:

1. Prvky ANN jsou buď maximálně aktivní (mají výstup blízký 1), nebo neaktivní (výstup blízký 0). Při tomto předpokladu každý člen ANN tvoří v podstatě booleovské pravidlo. Otázkou extrakce pravidel je najít kombinace vstupů, při kterém je takovéto „pravidlo“ splněno (rovno 1).
2. Trénovací algoritmus ANN nesmí významně měnit smysl jednotlivých prvků.

První metodu nazývanou „Subset algorithm“ (algoritmus podmnožin) popsali Saito a Nakano (1988) [31] a Fu (1991) [9]. Metoda je nazývána metodou podmnožin, protože vyhledává podmnožiny vstupů, při kterých suma vstupů přesáhne hodnotu prahu, tedy výstup neuronu je blízký 1. Podle prvního předpokladu je každý vstup blízký 1, proto lze při testování použít pouze sumu vah vstupů. Pro každou takovou podmnožinu vytvoří pravidlo ve formě $A \wedge B \wedge ... \rightarrow Y$, kde *A*, *B*,... jsou vstupy neuronu a *Y* výstup neuronu.

Na základě prvního předpokladu můžeme zjednodušit přenosovou funkci neuronu z původního tvaru (rovnice 2.4) na rovnici 2.6.

$$a_i = f\left(\sum_j \left(w_{i,j} \cdot a_j\right) - \theta_i\right) \tag{2.4}$$

$$f(x) = \frac{1}{1+e^{-sx}} \tag{2.5}$$

$$a_i = f\left(\sum_{\{j \mid a_j \approx 1\}} \left(w_{i,j}\right) - \theta_i\right) \tag{2.6}$$

kde a_i je vstup neuronu,

$w_{i,j}$ váha spoje neuronu *j* k neuronu *i*,

θ_i je práh neuronu,

s je parametr určující tvar přenosové funkce.

Problémem tohoto algoritmu je jeho efektivita, neboť počet extrahovaných pravidel roste s mocninou počtu vstupů do neuronu. Saito a Nakano [31] navrhli metodu omezení počtu antecedentů v pravidle, to však vede k problémům s použitím takovýchto pravidel při reálných problémech [39].

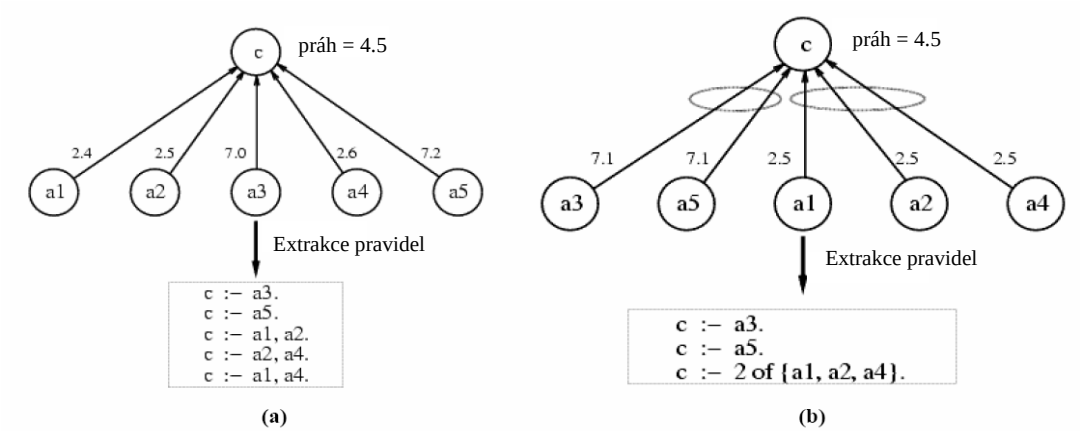
2.3.4 Algoritmus M of N

Geoffrey G. Towell a Jude W. Shawlik (1993) [39] upravili metodu Subset tak, že výrazně snižuje počet extrahovaných pravidel. Používají k tomu metodu „M of N“. Ta spočívá v zápisu pravidel tímto způsobem:

$$if\ \left(M\ z\ následujících\ N\ antedcedentů\ je\ pravdivých\right) tak ... \tag{2.7}$$

Př.: if 3 z {A, B, C, not(E)} tak Y

Pro seskupování vstupů s podobnými váhami využívají shlukové analýzy [45]. Extrakci pravidel tímto algoritmem ukazuje obrázek 7.



Obrázek 7: Extrakce pravidel z neuronové sítě [45]

2.3.5 Algoritmus RuleNet

RuleNet [29], [1] je třívrstvá neuronová síť mapující vstupní řetězec na výstupní řetězec, ze které jsou extrahována symbolická pravidla typu podmínka – akce. Algoritmus byl předveden na problému mapování jednoho řetězce na druhý. Pravidla představují vlastnost nebo kombinaci vlastností v každém vstupním řetězci na akci, která se má provést s každým symbolem výstupního řetězce.

ANN se skládá ze vstupní vrstvy, vrstvy podmínkových neuronů a výstupní vrstvy. Váhy spojující vstupní vrstvy s podmínkovými neurony reprezentují podmínkovou část pravidla, na kterou má být pravidlo naučeno. Váhy spojů z podmínkového neuronu zajišťují existenci pouze jedinečné vazby mezi i-tým znakem vstupního řetězce a j-tým znakem výstupního řetězce.

Pravidla jsou extrahována dekompilací vah vstupujících do podmínkového neuronu a vah mezi podmínkovou vrstvou a výstupní vrstvou. Každý prvek podmínkové části pravidla vytvoří za pomoci matice výstupních vah ANN pravidlo mapující vstupní znak řetězce na výstupní. Jako učicí algoritmus byla použita metoda Conectionst scientist game autorů Jakobse a spol. [15]. Tato metoda je založená na cyklickém zvyšování přesnosti pravidel a následné opětovné inicializace neuronové sítě těmito pravidly před procesem učení. Cyklus probíhá ve třech krocích:

1. Naučit síť na trénovací vzory.
2. Extrahovat symbolická pravidla ze sítě podle síly vah mezi neurony.
3. Vložit pravidla zpět do sítě.

Tento cyklus končí, když výsledná báze pravidel dostatečně reprezentuje danou problematiku. Výhodou tohoto přístupu je, že zde nedochází ke kombinatorickému prohledávání stavového prostoru, jako u jiných dekompozičních algoritmů, neboť jsou pravidla tvořena prozkoumáváním dvojicí vah (podmínková váha – výstupní váha). Nevýhodou algoritmu je, že byl zkonstruován pouze pro detekci vztahů mezi řetězci a postrádá tak na obecnosti. [24]

2.3.6 Algoritmus RULEX

RULEX [10] je metoda extrakce pravidel, která využívá způsob konstrukce a chování konsekventu speciálního typu neuronové sítě CEBP – constrained error back-propagation. Tato síť je druhem ANN s lokální odezvou, která aproximuje a klasifikuje podobně jako sítě s RBF neurony.

Skryté neurony sítě jsou lokálně citlivé neurony (LRU – Local Responsive Unit), které rozdělují trénovací data do oddělených regionů tvořících vrcholy ve stavovém prostoru sítě. Každý vnitřní neuron zastupuje podobně jako RBF, jeden vrchol. Vnitřní neuron je složen z množiny „hřebenů“, jeden hřeben pro jednu dimenzi vstupního prostoru. Hřeben je aktivován, pouze pokud vstupní hodnota sítě je uvnitř aktivní oblasti hřebenu. Výstup LRU je realizován prahovanou sumou aktivací jednotlivých hřebenů neuronu. Má-li LRU klasifikovat vstupní vektor do nějaké třídy, musí všechny složky vstupního vektoru ležet uvnitř aktivních oblastí příslušných hřebenů.

Z každého neuronu je potom extrahováno pravidlo ve formě:

If hřeben 1 *aktivní* a

hřeben 2 *aktivní* a

...

hřeben *N aktivní*,

then vzor patří do třídy $Class_i$.

Aktivní oblasti hřebenů mohou být počítány z jejich středu, šířky a kroku (C_i, B_i, k_i) v každé dimenzi. Je tedy možné přepsat pravidlo pomocí těchto parametrů takto:

If $c_1 - b_1 + 2k_1^{-1} \le x_1 \le c_1 + b_1 - 2k_1^{-1}$ a

$c_2 - b_2 + 2k_2^{-1} \le x_2 \le c_2 + b_2 - 2k_2^{-1}$ a

... a

$c_N - b_N + 2k_N^{-1} \le x_N \le c_N + b_N - 2k_N^{-1}$,

then vzor x patří do třídy $Class_i$.

Algoritmus RULEX obsahuje také mechanizmy pro odstranění přebytečných antecedentů pravidel, vytváření negativních antecedentů a slučování více konkrétních pravidel do jednoho obecnějšího. Algoritmus vytváří pravidla interpretací vektorů vah jako pravidel, a tím značně snižuje výpočetní náročnost oproti jiným dekompozičním technikám.

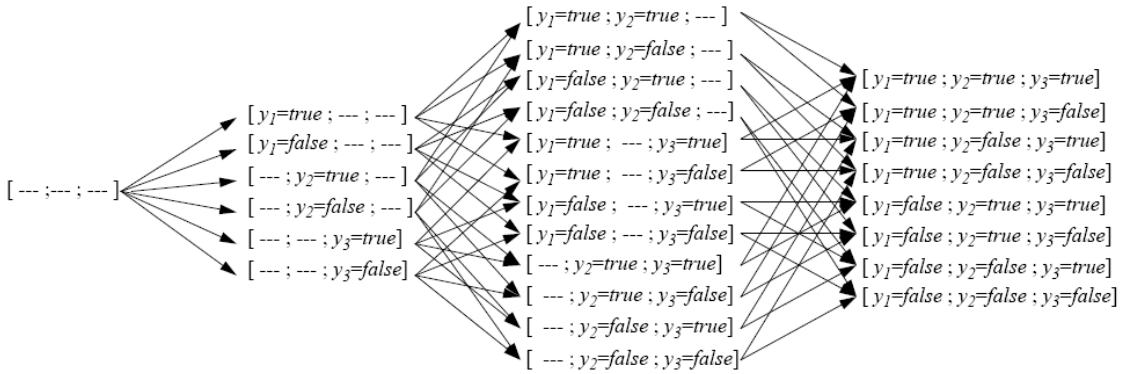
2.4 Extrakce pravidel použitím pedagogického přístupu

Kapitola popisuje extrakci pravidel metodami, které se dívají na ANN jako na černou skříňku a popisují chování ANN jako celku nezávisle na její struktuře a vnitřních funkcích neuronů. Metody analyzují chování výstupů ANN v závislosti na systematických změnách hodnot vstupů a vytvářejí pravidla modelující toto chování .

2.4.1 VI analýza

VI analýza (Validity Interval analysis) [36] je algoritmus tvořící pravidla na základě vyhodnocování intervalů platnosti jednotlivých antecedentů pravidla příslušejících určité klasifikační třídě.

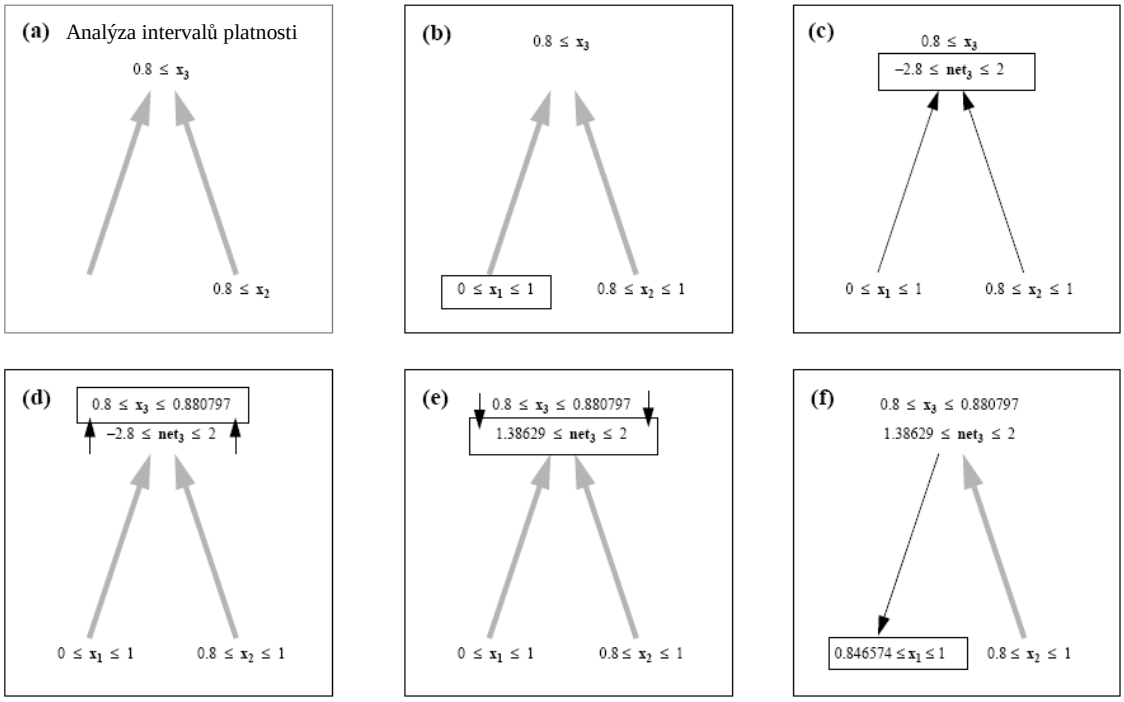
Pravidla jsou vytvářena na základě systematického prohledávaní stavového prostoru vstupních vzorů pro každý jeden cílový uzel ANN (viz. obrázek 8).



Obrázek 8: Graf pravidel pro tři vstupní binární atributy y_1,y_2,y_3 [36]

Pravidla v grafu na obrázku 8 jsou orientována od nejobecnějších ke specifickým (z leva doprava). V tomto směru je také graf prohledáván. Každé pravidlo je testována VI analýzou, a pokud správně reprezentuje klasifikační třídu, lze odstranit všechna následující více specifická pravidla a nahradit je tímto obecným. Tímto způsobem se výrazně zjednodušuje prohledávaný prostor pravidel.

Vlastní analýza je založena na šíření intervalů platnosti antecedentů pravidla (tedy vstupů neuronové sítě) skrz ANN a ověřování konzistentnosti vůči předpokládaným intervalům platnosti konsekventu (výstupu sítě) (obrázek 9). Každému vstupu jsou přiřazeny intervaly platnosti vstupních hodnot (obrázek 9: a, b). Tyto intervaly jsou jako celek šířeny neuronovou sítí a jsou vypočítány (z jejich hraničních hodnot) intervaly aktivace neuronu. Tyto intervaly aktivace jsou transformovány přes vnitřní funkci neuronu (autor uvažuje sigmoidu) a upřesňují interval platnosti konsekventu (obrázek 9: c). Na základě nového intervalu konsekventu jsou upřesněny zpětným šířením tohoto intervalu přes neuron intervaly platnosti antecedentů (obrázek 9: e, f). Situaci znázorňuje obrázek 9.



Obrázek 9: VI analýza [36]

Pokud neexistuje průnik mezi intervaly konsekventu a transformovaným intervalem aktivace neuronu, je pravidlo vyhodnoceno jako nekonzistentní.

Nedostatkem tohoto algoritmu je, že může selhat při řešení komplexních problémů, neboť analýza nebude schopna najít příslušné intervaly platnosti antecedentů [36].

2.4.2 Extrakce pravidel jako učení

Extrakce pravidel jako učení (Rule extraction as learning) [6] využívá k tvorbě pravidel testování vstupního prostoru a klasifikaci příslušných vzorů.

Algoritmus využívá dvě základní komponenty, a to Examples (příklady) a Subset (podmnožiny). Postup tvorby pravidel je ukázán v následujícím pseudo kódu [6].

*/*inicializace pravidel pro každou třídu*/*
pro každou třídu c
R_c *:= 0*
opakovat:
  e *:= Examples()*
  c *:= kasifikace(*e*)*
  jestliže e *není pokryta* R_c *pak*
    */*vytvoř nové pravidlo*/*
    r *:= konjunktivní pravidlo tvořené z* e
    pro každý antecedent r_i *z* r
    r´ *:=* r *s vynecháním* r_i
    *jestliže Subset(*c,r´*) = true pak* r *:=* r´
  R_c *:=* $R_c \vee$ r
dokud není dosažena zastavovací podmínka

Algoritmus cyklicky volá metodu Examples, která vytváří vstupní vektory pro ANN. Využívá k tomu data z trénovacích vzorů. Pokud již vzory nejsou k dispozici náhodně ohodnocuje vstupy. ANN klasifikuje vstupy do některé ze tříd, pokud tato třída nepokrývá daný vstupní vektor, vytvoří se nové pravidlo ze vstupního vektoru. Postupným vynecháváním jednotlivých konsekventů pravidla a testováním (metoda Subset), zda stále správně pokrývá danou třídu, se vytvoří co nejobecnější pravidlo (bez nepodstatných antecedentů).

Metoda Subset vrací *true*, když všechny instance pokryté pravidlem *r* jsou klasifikovány jako třída *c*, jinak vrací *false*. Jde o podobnou operaci jako testování pravidel u dekompozičních metod. Autoři uvádějí dvě možné implementace této metody, pomocí dekompozice nebo VI analýzy. Výběr této metody závisí na konkrétní řešené problematice.

Výhodou algoritmu je, že je zcela nezávislý na struktuře ANNa nevyžaduje speciální trénovací metody. Je také méně výpočetně náročný než dekompoziční přístupy. Nevýhodou této metody je, že stochasticky vytváří vstupní vektory, ze kterých jsou odvozovány pravidla a může tak trvat velmi dlouho než algoritmus dospěje k pravidlům s potřebnou obecností.

2.5 Algoritmus Re-RX

Rekurzivní algoritmus Re-RX (Recursive Rule eXtraction) [33] vytváří produkční pravidla rekurzivním trénováním ANN a extrakcí pravidel.

Algoritmus Re-RX pracuje s dopřenou ANN bez zvláštních požadavků na její strukturu a s diskrétními i spojitými vstupními daty. Extrakce pravidel je prováděna ve čtyřech krocích.

V prvním kroku je ANN natrénována na všechny vstupní vzory. Následně jsou odstraněny nevýznamné spoje (spoje s váhou blízkou nule) a neurony sítě, a to i vstupní. Autoři ve své dřívější práci uvádějí takovýto algoritmus čištění ANN [34]. Ve druhém kroku jsou extrahována pravidla pro diskrétní vstupy algoritmem C4.5 a je ohodnocena jejich přesnost vzhledem k testovacím vzorům. Ve třetím kroku je vybráno pravidlo s chybou klasifikace příslušných vzorů větší, než je požadovaný limit (může být i nula, tj. pravidla jsou sestavena s co největší přesností vůči původní ANN, zvyšuje se tak ale významně počet pravidel). Toto pravidlo je odstraněno a správně klasifikované vzory příslušející tomuto pravidlu jsou použity k natrénování nové (třívrstvé) neuronové sítě. Vstupními parametry této nové sítě jsou vstupy příslušných vzorů a vstupy se spojitou vstupní veličinou. Na novou ANN jsou uplatněny předchozí body algoritmu a původní nevyhovující pravidlo je nahrazeno novými pravidly získanými z nové ANN. Tento proces se opakuje, dokud nejsou pravidly pokryty všechny vstupní parametry původní ANN. V posledním kroku jsou separovány do tříd vstupy se spojitými vstupními hodnotami jako zpřesnění příslušných diskrétních pravidel. Pro tyto vstupy jsou spojité hodnoty vzorů klasifikovány do hyperprostorů regresními metodami a SVM[1]. Následně je celý postup volán pro ostatní nepřesná pravidla, do konečného upřesnění modelu. Pravidla pro spojité vstupní veličiny jsou tvořena ve formě:

If vstup N $\leq$ hodnota1 *and* vsup N-1 $\geq$ hodnota2 .... *then class N*.

[1] Support Vector Machine

Autoři ukazují, že touto metodou dosahují vyšší přesnosti extrahovaných pravidel než při použití jiných metod extrakce.

2.5.1 Algoritmus REANN

Rule Extraction from Artificial Neural Networks (REANN) [21] je algoritmus pro extrakci produkčních pravidel z třívrstvé dopředné ANN typu back-propagation.

Neuronová síť je zde tvořena konstruktivním algoritmem, který určuje počet skrytých neuronů sítě. Po naučení ANN na trénovací vzory jsou odstraněny nepodstatné spoje a vstupy sítě. Následně jsou výstupy vnitřních uzlů disketrtizovány heuristickým algoritmem a pomocí těchto diskrétních tříd jsou vytvořena pravidla mapující vstupy na výstupy. Na takto vygenerovaná pravidla je použit algoritmus čištění pravidel, který seskupuje pravidla podle klasifikačních tříť a v rámci těchto tříd nahrazuje specifická pravidla obecnějšími. Proces pokračuje, dokud nejsou pokryty všechny vzory některým z pravidel.

2.6 Shrnutí

Reprezentace znalostí pravidly (pravidlovými sítěmi) a neuronovými sítěmi mají své výhody stejně jako své nevýhody. V případě těchto dvou reprezentací jsou jejich vlastnosti většinou komplementární. Vzniklo proto několik algoritmů, které se snaží kombinací obou maximálně využít jejich cenné vlastnosti. Tyto algoritmy jsou založeny na extrakci pravidel z naučené neuronové sítě, inicializace ANN pravidly, nebo kombinací obou těchto přístupů. První skupinou těchto algoritmů jsou dekompoziční přístupy analyzující vnitřní stavy neuronů v ANN.

Fu nabízí metodu, která mapuje pravidla do neuronové sítě. Zachovává sémantiku pravidel a umožňuje modelování neurčitosti. Hlavní výhodou je jednoduchá zpětná extrakce pravidel z naučené sítě. Nevýhodou tohoto algoritmu je učící mechanizmus, neboť konjunktivní vrstvy nejsou diferencovatelné a tedy nelze použít klasické učící algoritmy. Metoda také nabízí identifikaci a odstranění nekorektních pravidel, není ovšem stoprocentní. Fu ve své práci také popisuje algoritmus vytváření nových pravidel. Nikdy ho ovšem neimplementoval [5].

Towell a Shavlik navrhli metodu KBANN využívající pravidla k inicializaci sítě. Metoda umožňuje použití standardních algoritmů učení a vyniká rychlostí pro malé sítě. Oproti standardním ANN lépe zpracovává neurčitost na vstupech. Nevýhodou je ztráta sémantiky pravidel v důsledku přidání vazeb mezi neurony. To dělá případnou extrakci pravidel extrémně obtížnou [5].

Algoritmy extrakce pravidel z ANN spočívají na vstupech neuronu, které v součtu vah překonají jeho práh. Z příslušných vstupů je vytvořeno pravidlo. Algoritmus podmnožin dává klasická pravidla ve formě implikace antecedentů na konsekvent. Nevýhodou je zhoršení přesnosti oproti ANN ze které byly pravidla extrahována a obrovský počet výstupních pravidel. Algoritmus je exponenciální, zatímco algoritmus M of N je přibližně kubický. M of N poskytuje pravidla se srovnatelnou přesností s ANN a výrazně méně pravidel než předchozí. Nevýhodou tohoto algoritmu je špatná srozumitelnost výstupních pravidel a tím i jejich případné zpracování. Dalším problémem obou metod je způsob nastavování prahů neuronů [39].

Algoritmus RuleNet využívá k extrakci pravidel cyklického učení neuronové sítě, extrakce pravidel na základě analýzy vah neuronů a opětné inicializace ANN extrahovanými pravidly. Dochází tak k postupnému upřesňování pravidlového modelu. Výhodou algoritmu je, že nedochází ke kombinatorickému prohledávání stavového prostoru. Algoritmus byl vytvořen pouze pro detekci vztahů mezi řetězci, nelze jej tedy obecně uplatnit [24].

Algoritmus RULEX využívá k extrakci pravidel speciální typ neuronové sítě CEBP. Analýzou vah neuronové sítě vytváří pravidla, která popisuje aktivačními funkcemi aktivních neuronů pro předložený vzor. Výhodou algoritmu je nižší výpočetní náročnost [10], cenou je ovšem srozumitelnost takto extrahovaných pravidel.

Ve druhé skupině je možné jmenovat pedagogické algoritmy, které analyzují chování neuronové sítě jako celku.

Typickým představitelem této skupiny je VI analýza, která tvoří pravidla systematickým prohledáváním vstupního stavového prostoru ANN pro každý výstupní neuron a analýzou intervalů platnosti jednotlivých antecedentů pravidla příslušejících určité klasifikační třídě. Výhodou algoritmu je, stejně jako u ostatních pedagogických metod, že je zcela nezávislý na struktuře a učicím algoritmu ANN. Nedostatkem algoritmu je, že může selhat pro komplexní domény, neboť nemusí být nalezeny potřebné intervaly platnosti antecedentů [36].

Algoritmus Rule extraction as learning vytváří pravidla z naučené ANN klasifikováním vstupních vzorů do tříd a odstraňováním nepodstatných antecedentů (vstupních proměnných ANN) z pravidla. Nevýhodou algoritmu je, že stochasticky vytváří vstupní vektory, ze kterých jsou odvozovány pravidla, a může tak velmi dlouho, než algoritmus dospěje k pravidlům s potřebnou přesností.

Algoritmus Re-RX vytváří pravidla rekurzivním trénováním ANN a extrakcí pravidel. Pravidla extrahuje algoritmem C4.5 a vyhodnocuje přesnost vzhledem k testovacím vzorům. Pro nepokryté vzory vytvoří novou ANN a natrénuje ji nepokrytými vzory. Původní chybná pravidla jsou nahrazena extrakcí pravidel z nové ANN. Výhodou algoritmu je, že poskytuje produkční pravidla s vekou srozumitelností. Nevýhodou je velká výpočetní náročnost algoritmu způsobená několikanásobným trénováním ANN.

Současné metody tvorby bází znalostí využívající kombinaci pravidel a neuronových sítí mají jednu společnou nevýhodu. Touto nevýhodou je vlastní princip tvorby a ladění pravidlových bází znalostí extrakcí pravidel z naučené neuronové sítě. Nebudeme-li se zabývat faktem, že tento proces je značně výpočetně náročný [37], jde o proces, při kterém je vytvářen model reálné domény neuronovou sítí a následně je tato neuronová síť modelována pravidly, což snižuje přesnost výsledného pravidlového modelu vůči pozorované reálné doméně. Je totiž vytvářen model modelu. Mnohé ze současných algoritmů extrakce pravidel také neposkytují pravidla v příliš srozumitelné formě (např. M of N, popis aktivačními oblastmi neuronů, fuzzy množiny, atd.). Srozumitelný popis činnosti ANN byl původní impulz pro tvorbu těchto algoritmů. Výhodou těchto algoritmů je, že dokáží vytvořit pravidlové báze znalostí analýzou dat bez jakýchkoliv znalostí o dané doméně.

Tato práce se zabývá vývojem algoritmu pro tvorbou a automatické ladění pravidlových bází znalostí pro expertní systémy, které mohou být nasazeny v kritických a bezpečných aplikacích. Je tedy kladen značný důraz na srozumitelnost pravidel v bázi znalostí, tak aby mohly být vysvětleny postupy a závěry expertního systému. V drtivé většině případů stávajících systémů jsou báze znalostí vytvářeny na základě znalostí a dat od expertů. Není tedy nezbytně nutná přímá analýza dat.

Cílem této práce je najít metodu tvorby a ladění báze znalostí pro expertní systémy pomocí informací získaných od expertů s důrazem na srozumitelnost výsledné báze znalostí.

3 Cíle disertace

Cílem této disertační práce je najít metodu automatického ladění vah pravidel v bázi znalostí expertních systémů a eliminovat tak vysokou časovou náročnost tvorby těchto bází znalostí. Důraz je kladen na vysokou srozumitelnost báze znalostí tak, aby mohly být uplatněny vysvětlovací mechanizmy.

Pro návrh algoritmu automatického nastavování vah pravidel je nutné zajistit vhodnou strukturu báze znalostí, způsob vyjadřování a zpracování neurčitosti, vhodnou metodu získávání znalostí od experta, tedy tvorbu vzorů a stanovit kritérium učení báze znalostí pro vyjádření chyby v průběhu adaptace vah pravidel.

Cíl 1

Najít řešení automatického ladění vah pravidel pro jednovrstvé báze znalostí expertního systému (NPS32). Pro ladění využít data a znalosti poskytované přímo expertem.

Cíl 2

Najít analytické řešení automatického ladění vah pravidel pro vícevrstvé, libovolně strukturované pravidlové báze znalostí expertních systémů.

Cíl 3

Algoritmy ladění vah pravidel v bázi znalostí implementovat do prázdného expertního systému (shell).

Cíl 4

Analytické řešení ladění vah pravidel v bázi znalostí ověřit vytvořením reálné funkční báze znalostí a provést srovnání s bází znalostí tvořenou znalostním inženýrem a expertem.

4 Ladění vah báze znalostí s evolučním algoritmem

Kapitola popisuje algoritmus ladění vah pravidel v bázích znalostí expertního systému NPS32. Je zde popsán vlastní matematický aparát expertního systému NPS32, matematická reprezentace a struktura báze znalostí, metoda učení, optimalizace vah pravidel báze znalostí na základě stanovené kriteriální funkce a experimentální ověření funkce algoritmu. Optimalizace vah pravidel je prováděna diferenciálním evolučním algoritmem. V závěru kapitoly je proveden formální důkaz, že matematický aparát expertního systému NPS32 nelze použít pro gradientní optimalizaci u vícevrstvých bází znalostí.

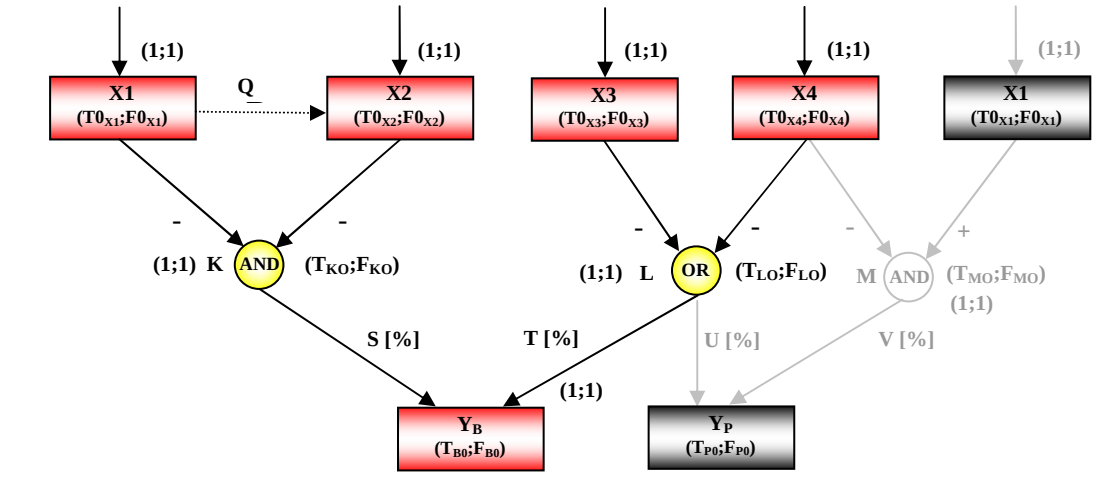
Metody založené na učení neuronové sítě a zpětné extrakci pravidel poskytují obrovská množství pravidel nebo pravidla v nepříliš srozumitelné formě a jsou zatíženy nepřesnostmi při převodu reálné domény na ANN a z ANN na pravidla. Tato kapitola a kapitola 5 popisují možnosti přímého nastavování vah ve struktuře pravidel báze znalostí na základě předem daných vzorů. Předejde se tak složité extrakci naučených pravidel z neuronové sítě při zachování učení z dat a velmi dobré srozumitelnosti báze znalostí.

Princip ladění vah pravidel jednovrstvých kontextově vázaných bází znalostí byl implementován do stávajícího diagnostického expertního systému NPS32 [44]. Správnost funkce byla ověřena naučením dvou bází znalostí. Jako učící algoritmus je zde použit diferenciální evoluční algoritmus popsaný v kapitole 4.4.

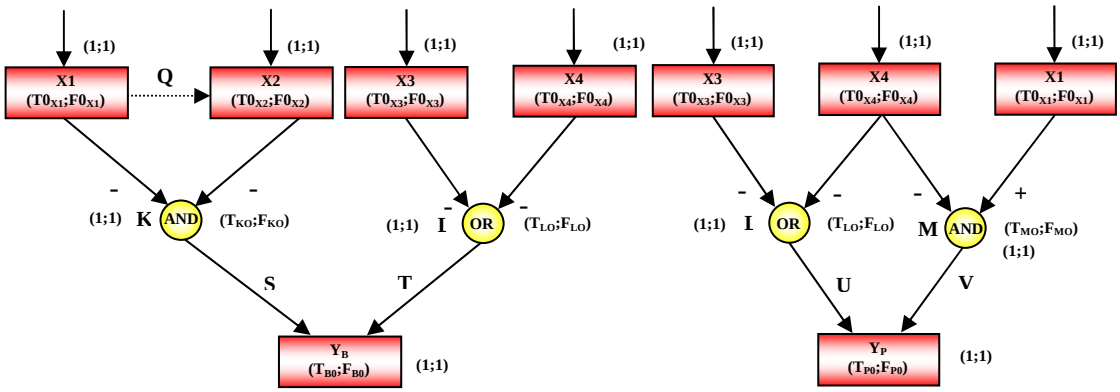
4.1 Báze znalostí expertního systému NPS32

Báze znalostí expertního systému NPS32 je reprezentována orientovaným grafem pravidel ohodnocených neurčitostmi. Struktura báze je tvořena jednou vrstvou pravidel vázaných na cílové uzly agregující informaci z pravidel. Každé pravidlo si můžeme představit jako samostatný orientovaný graf s několika kořenovými uzly (dotazovatelné uzly) a jedním listovým uzlem (cílový uzel). Cílové uzly agregují informaci z pravidel do něj vstupujících, které snižují či zvyšují pravděpodobnost hypotézy cílového uzlu. Informační vliv jednotlivých pravidel je dán silou vazby pravidla na cíl a změnou hodnoty pravidla.

Každý jeden dotazovatelný uzel může ovlivňovat více pravidel (viz. obrázek 10, vstup X4, pravidla L a M). Báze znalostí s tímto druhem vazeb je vhodné, pro zjednodušení výpočtu, rozdělit na podstromy pravidel vázaných na jeden cílový uzel, tedy jeden cílový uzel se všemi příslušnými dotazovatelnými uzly vázanými pravidly. Situaci ilustruje obrázek 11. Bázi znalostí rozdělujeme tak, že pro každý cílový uzel zkopírujeme všechna pravidla do něj vstupující. Těmto pravidlům pak přiřadíme všechny vstupní uzly které jej ovlivňují. Vzniknou nám tak samostatné podstromy grafu s možnou duplikací vstupních dotazovatelných uzlů.



Obrázek 10: Struktura jednoho vybraného pravidla v bázi znalostí



Obrázek 11: Ukázka rozdělení báze znalostí na podstromy

Rozdělením báze znalostí na jednotlivé podstromy pravidel zjednodušíme úlohu hledání globálního řešení na několikanásobné (podle počtu cílů) hledání vektoru vah pravidel ovlivňující jediný cílový uzel. Zamezíme tak ovlivnění výpočtu vah pravidel výstupními chybami z jiných z podstromů.

4.2 Matematický aparát expertního systému NPS32

Přestože je expertní systém založen na práci s pravděpodobností, většinou se v programu nevyskytují typická vyjádření pravděpodobnosti pomocí proměnných nabývajících hodnot z intervalu 0 – 100 % (P), resp. 0 – 1 (p), ale jsou použita vyjádření pravděpodobnosti pomocí dvou hodnot (T, F). Zavedení dvou hodnot každému uzlu [7] přispělo k zjednodušení výpočetních vztahů a zároveň zpřístupňuje informaci o *spornosti některého tvrzení* a lépe *vyjadřuje nepřítomnost informace*.

Chápeme-li výraz (1-T) jako míru důvěry v pravdivost tvrzení a naopak výraz (1-F) jako míru důvěry v nepravdivost tvrzení, pak dvojce (T;F) [7]:

(0;1) odpovídá pravdivému tvrzení,

(1;0) odpovídá nepravdivému tvrzení,

(1;1) znamená nepřítomnost jakékoliv informace,

(0;0) znamená spor v bázi znalostí nebo v odpovědích uživatele.

Základní matematický aparát pro výpočty s hodnotami T a F je dán následujícími vztahy a operátory:

$$T = \frac{F(1-p)}{p} \tag{4.1}$$

$$F = \frac{pT}{1-p} \tag{4.2}$$

kde p je pravděpodobnost. Pro T, $F \in$ <0, 1> je $p \in$ <0, 1>.

Jestliže je p < 0,5 (50%) volíme T maximální (tj. T=1) a F dopočteme pomocí vztahu (4.2) a naopak pokud je p > 0,5 (50%) volíme F maximální (tj. F=1) a T dopočteme pomocí vztahu (4.1).

Pravděpodobnost uzlu je dána:

$$P = p \cdot 100 = \frac{F}{T+F} \cdot 100 \quad [\%] \tag{4.3}$$

Negace („–“):

$$not(T,F) = (F,T) \tag{4.4}$$

Konjunkce („&“):

$$(T_1,F_1)and(T_2,F_2) = (\max(T_1,T_2),\min(F_1,F_2)) \tag{4.5}$$

Disjunkce („|“):

$$(T_1,F_1)or(T_2,F_2) = (\min(T_1,T_2),\max(F_1,F_2)) \tag{4.6}$$

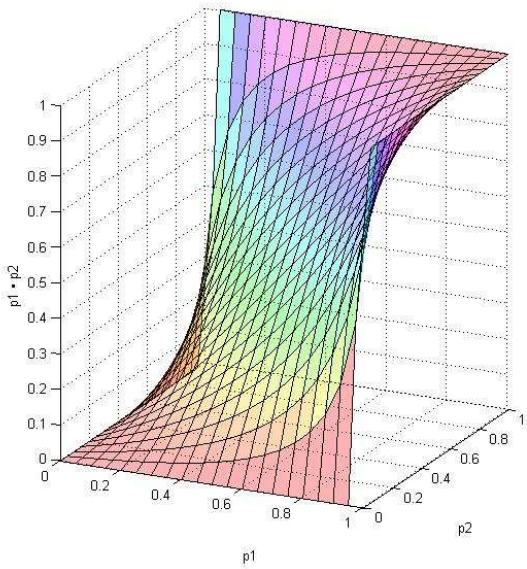
Skládání s odpovědí uživatele („·“):

$$(T_1;F_1)\cdot(T_2;F_2) = (T_1 \cdot T_2; F_1 \cdot F_2) \tag{4.7}$$

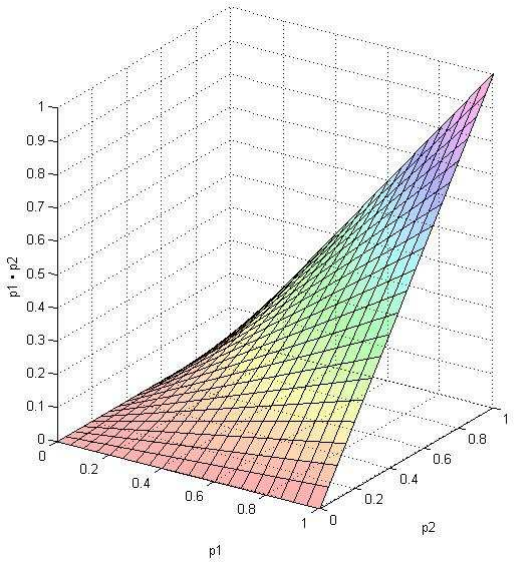
Vztah vyjadřuje vliv odpovědi uživatele na neurčitost zodpovězeného uzlu.

kde $(T_1; F_1)$ je původní neurčitost uzlu

$(T_2; F_2)$ je vliv neurčitosti odpovědi.



Stavový prostor skládání neurčitosti podle vztahu (4.7)



Stavový prostor skládání klasické pravděpodobnosti

Obrázek 12: Stavové prostory skládání neurčitostí

Porovnáním grafů z obrázku 12 je zřejmý silnější vliv uživatelské odpovědi na dotazovatelný uzel při použití dvou hodnot neurčitosti (matematický aparát NPS32), než by tomu bylo u klasické pravděpodobnosti. Také je zde patrný rychlejší přechod mezi důvěrou (hodnota pravděpodobnosti $p1 \cdot p2 = 1$) a nedůvěrou ve výsledek (hodnota pravděpodobnosti $p1 \cdot p2 = 0$). To má za následek snížení citlivosti uzlů s malým vlivem odpovědi a nemůže tak dojít k zásadní změně důvěry či nedůvěry hypotézy zastoupenou uzlem, který je ovlivněn odpovědí se zásadním vlivem [18].

Skládání silou vazby („→“):

Tento vztah popisuje vliv odpovědi uživatele na neurčitost všech zbývajících uzlů, se kterými je zodpovězený uzel vázán pomocí pravidel. Zabezpečuje tedy šíření informace v systému [20].

$$(T;F) = (T_1/T_0\,;F_1/F_0) \xrightarrow{\alpha} (T_2;F_2) \tag{4.8}$$

$$T = \left(\frac{w*T_1 - w + 1}{w*T_0 - w + 1}\right) * T_2, \tag{4.9}$$

$$F = \left(\frac{w*F_1 - w + 1}{w*F_0 - w + 1}\right) * F_2, \tag{4.10}$$

kde w je vyjádření síly vazby,

(T; F) je aktuální neurčitost uzlu, do kterého pravidlo vstupuje,

(T_0; F_0) je původní neurčitost uzlu, ze kterého pravidlo vystupuje,

(T_1; F_1) je neurčitost uzlu po odpovědi, ze kterého pravidlo vystupuje,

(T_2; F_2) je původní neurčitost uzlu do kterého pravidlo vstupuje.

Tím jsou definovány základní vztahy pro práci s hodnotami T a F. Již na první pohled je zřejmé, že zavedením dvou hodnot se nám rozšířily možnosti expertního systému, mimo jiné v podobě poskytnutí informace o spornosti kteréhokoliv tvrzení – hodnoty T a F jsou blízké nule. Je zde také zřejmé, že k ovlivňování výstupní hodnoty cílového uzlu dochází změnou hodnoty pravidla do něj vstupujícího, nikoliv tedy přímo novou hodnotou pravidla.

4.2.1 Realizace násobných vazeb v cílovém uzlu

Z příkladu grafického vyjádření vazeb báze znalostí (obrázek 10) je patrná nutnost exaktního vyjádření vztahu mezi vektorem vstupů $\overline{X}$ a výslednou hodnotou pravděpodobnosti hypotézy Y.

Ze vztahů (4.5), (4.6) a (4.7) je možné vypočíst hodnotu pravidel K a L (obrázku 10) po zodpovězení dotazovatelných uzlů *X1* až *X4*. Označme tento vztah Φ. Tedy

$$(T_K, F_K) = \Phi\left(P_{od}\big|_{X1}, P_{od}\big|_{X2}\right), \tag{4.11}$$

$$(T_L, F_L) = \Phi\left(P_{od}\big|_{X3}, P_{od}\big|_{X4}\right). \tag{4.12}$$

Ze vztahů (4.9) a (4.10) plyne vliv pravidla na výsledek pro pravidlo K takto:

$$T_1 = \left(\frac{w_K * T_{K1} - w_K + 1}{w_K * T_{K0} - w_K + 1}\right) * T_0, \tag{4.13}$$

$$F_1 = \left(\frac{w_K * F_{K1} - w_K + 1}{w_K * F_{K0} - w_K + 1}\right) * F_0, \tag{4.14}$$

kde w_K je vyjádření síly vazby,
(T_{K1}; F_{K1}) je pravděpodobnost pravidla po odpovědi dotazovatelných uzlů,
(T_{K0}; F_{K0}) je původní pravděpodobnost pravidla před odpovědí dotazovatelných uzlů,
(T_0;F_0) je původní pravděpodobnost uzlu Y, do kterého pravidlo vstupuje,
(T_1;F_1) je aktuální pravděpodobnost uzlu Y, do kterého pravidlo vstupuje.

Pro pravidlo *L* již ve výpočtu musíme uvažovat vliv pravidla *K* na výsledný uzel Y.

$$T_2 = \left(\frac{w_L * T_{L1} - w_L + 1}{w_L * T_{L0} - w_L + 1} \right) * T_1, \tag{4.15}$$

$$F_2 = \left(\frac{w_L * F_{L1} - w_L + 1}{w_L * F_{L0} - w_L + 1} \right) * F_1, \tag{4.16}$$

kde w_L je vyjádření síly vazby,
(T_{L1}; F_{L1}) je pravděpodobnost pravidla po odpovědi dotazovatelných uzlů,
(T_{L0}; F_{L0}) je původní pravděpodobnost pravidla před odpovědí dotazovatelných uzlů,
(T_1;F_1) je původní pravděpodobnost uzlu Y po ovlivnění pravidlem *K*,
(T_2;F_2) je aktuální pravděpodobnost uzlu Y.

Vyjádříme-li tento rekurzivní výpočet jedním vztahem, dostáváme

$$T_1 = \left(\frac{w_K * T_{K1} - w_K + 1}{w_K * T_{K0} - w_K + 1} \right) \left(\frac{w_L * T_{L1} - w_L + 1}{w_L * T_{L0} - w_L + 1} \right) * T_0, \tag{4.17}$$

$$F_1 = \left(\frac{w_K * F_{K1} - w_K + 1}{w_K * F_{K0} - w_K + 1} \right) \left(\frac{w_L * F_{L1} - w_L + 1}{w_L * F_{L0} - w_L + 1} \right) * F_0, \tag{4.18}$$

Zobecněním tohoto vztahu dostáváme výsledný vztah zahrnující vliv libovolného počtu pravidel na výsledek

$$T = \left(\prod_{\forall k} \left(\frac{w_k * T_{k1} - w_k + 1}{w_k * T_{k0} - w_k + 1} \right) \right) * T_0 \tag{4.19}$$

$$F = \left(\prod_{\forall k} \left(\frac{w_k * F_{k1} - w_k + 1}{w_k * F_{k0} - w_k + 1} \right) \right) * F_0. \tag{4.20}$$

kde w_k je vyjádření síly vazby *k*-tého pravidla,
(T_{k1}; F_{k1}) je pravděpodobnost *k*-tého pravidla po odpovědi dotazovatelných uzlů,
(T_{k0}; F_{k0}) je původní pravděpodobnost *k*-tého pravidla před odpovědí dotazovatelných uzlů,
(T_0;F_0) je původní pravděpodobnost uzlu Y,
(T;F) je aktuální pravděpodobnost uzlu Y.

4.2.2 Vliv kontextové vazby

Kontextová vazba (vazba Q v obrázku 10) má v bázi znalostí pouze úlohu řízení inference a tedy nemá žádný vliv na výsledek. Tím dochází k podstatnému zjednodušení úlohy, je však nutné toto tvrzení dokázat.

Důkaz je proveden v souladu s obrázkem 10 pravidlem *K*.

Pokud má být tvrzení pravdivé, musí být výsledek při použití kontextové vazby shodný s výsledkem bez použití kontextové vazby.

Tj. $$Y\big|_{s\ vaybou} = Y\big|_{bez\ vazby}, \tag{4.21}$$

Je tedy potřeba odvodit výslednou hodnotu pro graf bez kontextové vazby.

Nemá-li být kontextová vazba splněna, musí platit

$$P\{X1\} \geq R, \tag{4.22}$$

kde R je hodnota kontextu.

Jelikož přepočet výsledku z pravidla je na kontextové vazbě nezávislý, je možné pro jednoduchost počítat pouze s hodnotami pravidla

$$(Tp, Fp) = \left(\max\left(T_1\big|_{P_{od}1}, ..., T_N\big|_{P_{od}N}\right), \min\left(F_1\big|_{P_{od}1}, ..., F_N\big|_{P_{od}N}\right)\right), \tag{4.23}$$

kde (Tp; Fp) je pravděpodobnost pravidla po odpovědi dotazovatelných uzlů, $\left(T_N\big|_{P_{od}N}, F_N\big|_{P_{od}N}\right)$ je pravděpodobnost N-tého dotazovatelného uzlu po odpovědi.

Pro pravidlo typu „OR“ je zaměněno ve vztahu min a max.

Není-li kontextová vazba splněna, bude výsledná hodnota pravidla

$$(Tp, Fp)\big|_{bez\ vazby} = \left(\max\left(T_1\big|_{P_{od}1}, T_{20}\right), \min\left(F_1\big|_{P_{od}1}, F_{20}\right)\right). \tag{4.24}$$

Bude-li kontextová vazba splněna, tedy

$$P\{X1\} \leq R, \tag{4.25}$$

bude výsledná hodnota pravidla

$$(Tp, Fp)\big|_{s\ vazbou} = \left(\max\left(T_1\big|_{P_{od}1}, T_2\big|_{P_{od}2}\right), \min\left(F_1\big|_{P_{od}1}, F_2\big|_{P_{od}2}\right)\right). \tag{4.26}$$

Při zachování stejných podmínek, tj. hodnota odpovědi v obou případech na uzel U_2 bude stejná, bude

$$T_2\big|_{P_{od}2} = T_{20},\ F_2\big|_{P_{od}2} = F_{20}. \tag{4.27}$$

Odtud, tedy z rovnic (4.24), (4.26) a (4.27) plyne

$$(Tp, Fp)\big|_{bez\ vazby} = (Tp, Fp)\big|_{s\ vazbou} \Rightarrow \quad Y\big|_{s\ vaybou} = Y\big|_{bez\ vazby} \tag{4.28}$$

Tvrzení je tedy platné. Plyne z něj, že pro výpočet vah pravidel není potřeba uvažovat kontextové vazby. Při tvorbě vzorů s kontexty však počítat musíme, neboť ovlivňují vstupní informaci (hodnoty dotazovatelných uzlů).

4.3 Metoda ladění vah pravidel v bázi znalostí

Obdobně jako u neuronových sítí, jsou váhy pravidla nastavovány na základě vzorů předkládaných expertnímu systému. Vzory mají formu požadované reakce systému na vstupní hodnoty, tedy vektor vstupních hodnot a požadovaná hodnota

cílového uzlu. Vzory jsou pro tento program vytvářeny expertem na základě jeho znalostí a zkušeností, mohou být ovšem získávány i metodami data-miningu z dat. Váhy pravidel v jednotlivých podstromech báze znalostí jsou nastavovány diferenciálním evolučním algoritmem na základě kriteriální funkce 4.34 odvozené níže.

Postup trénování báze znalostí je následující [46]:

1. Vytvoření struktury báze znalostí – expert / znalostní inženýr.
2. Rozdělení grafu na jednotlivá pravidla – program.
3. Vytvoření trénovacích vzorů – expert.
4. Výpočet vah pravidel báze – program.
5. Složení grafu – program.

Diferenciální evoluční algoritmus, použitý k optimalizaci vah pravidel, vytváří pro každý cílový uzel vektor vah pro příslušná pravidla. Váhy jsou vloženy do příslušného podstromu báze znalostí a je stanovena chyba klasifikace báze znalostí na základě kriteriální funkce. Tvorba vektorů vah pravidel a výpočet chyby klasifikace se opakuje do dosažení stanovené maximální chyby klasifikace nebo do dosažní maximálního počtu iterací.

4.3.1 Kriteriální funkce

Jako vhodná kriteriální funkce byla zvolena Euklidova vzdálenost mezi požadovanou hodnotou výstupu a hodnotu výstupu odpovídající příslušnému vzoru [41], [46]. Euklidova vzdálenost byla zvolena, neboť se v kritériu uplatňuje více pro velké odchylky výstupu od vzoru a méně pro malé odchylky, které mají při adaptaci vah pouze malý vliv. Malé odchylky způsobí pouze malé změny vah báze znalostí.

Mějme tedy množinu vzorů

$$M = \{\overline{X}_i, d_i\}, \quad i = 1,...,N, \tag{4.29}$$

kde $\overline{X}_i$ je vektor příslušných vstupních proměnných,

d_i je požadovaná hodnota výstupu,

i je pořadí vzoru,

N je počet učebních vzorů daného pravidla.

Každé pravidlo báze znalostí tvoří ohodnocený orientovaný graf realizující zobrazení z $R^N \rightarrow R^1$

$$\Psi: \quad \overline{X}_i \rightarrow Y_i, \tag{4.30}$$

kde N je délka, neboli počet prvků, vektoru $\overline{X}_i$.

Standardní Euklidova vzdálenost je definována jako

$$\varepsilon = (d_i - Y_i)^2, \tag{4.31}$$

která představuje míru či velikost odchylky výstupu grafu od požadované hodnoty pro daný vzor.

Kritérium naučení sítě definujeme jako střední hodnotu vzdálenosti ε přes všechny vzory M:

$$\mathrm{E}\{\varepsilon\} = \frac{1}{N}\sum_{\forall i}^{N}\left(d_i - Y_i\right)^2 . \tag{4.32}$$

Požadujeme tedy střední hodnotu odchylky přes všechny vzory minimální:

$$\mathrm{E}\{\varepsilon\} \rightarrow \min . \tag{4.33}$$

Dosazením rovnic (4.19), (4.20) do (4.3) a následně do (4.32) dostáváme kriteriální funkci odchylek výstupu grafu od požadované hodnoty pro všechny vzory

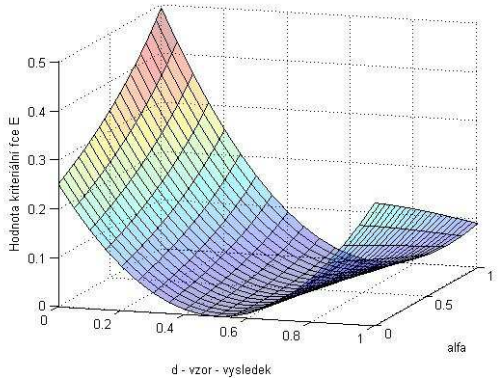
$$\mathrm{E}\{\varepsilon\} = \frac{1}{N}\sum_{\forall k \in M}\left(d_k - \frac{\left(\prod_{\forall k}\left(\frac{w_k * F_{k1} - w_k + 1}{w_k * F_{k0} - w_k + 1}\right)\right) * F_0}{\left(\prod_{\forall k}\left(\frac{w_k * T_{k1} - w_k + 1}{w_k * T_{k0} - w_k + 1}\right)\right) * T_0 + \left(\prod_{\forall k}\left(\frac{w_k * F_{k1} - w_k + 1}{w_k * F_{k0} - w_k + 1}\right)\right) * F_0} \right)^2 \tag{4.34}$$

kde w_k je vyjádření síly vazby k-tého pravidla,
(T_{k1}; F_{k1}) je pravděpodobnost k-tého pravidla po odpovědi dotazovatelných uzlů,
(T_{k0}; F_{k0}) je původní pravděpodobnost k-tého pravidla před odpovědí dotazovatelných uzlů,
(T_0;F_0) je původní pravděpodobnost uzlu Y,
(T;F) je aktuální pravděpodobnost uzlu Y,
M je počet učebních vzorů daného pravidla,
k je počet dotazovatelných uzlů příslušného pravidla. [41]

Tato funkce je v programu minimalizována diferenciálním evolučním algoritmem [35], [46]. Algoritmus nastavuje váhu w_k každého pravidla implikujícího cílový uzel Y. Vstupními parametry jsou vzory, tedy konstantní, uživatelsky definované, dvojce $\{\overline{X}_i, d_i\}$. Proces učení se zastaví, když změna výsledku nevykazuje po určitý počet iterací zlepšení.

Alternativou minimalizace kriteriální funkce je pro tento konkrétní případ využití některé z gradientních metod, protože se jedná o spojitou, obecně n-dimenzionální plochu.

Pro pravidlo s jedním antecedentem a počátečními neurčitostmi všech uzlů rovnými 50% je závislost hodnoty kriteriální funkce na váze *w* a hodnotě cílového uzlu ukázána na obrázku 13. Hodnota uživatelské odpovědi je 0,7.



Obrázek 13: Závislost hodnoty kriteriální funkce na hodnotě vzoru a váze alfa pro pravidlo s jedním antecedentem

4.4 Optimalizace vah pravidel diferenciálním evolučním algoritmem

Diferenciální evoluční algoritmus (DE) uvedl poprvé v roce 1996 autor Rainer Storn [35] jako univerzální populačně orientovaný optimalizátor.

Algoritmus diferenciální evoluce je iterativní algoritmus, který v jedné iteraci (generaci) zpracuje celou množinu potenciálních řešení - jedinců. Jedinec je reprezentován jako reálný vektor charakterizující pozici v prohledávaném stavovém prostoru. V našem případě jde o vektor vah pravidel příslušného podstromu. Noví jedinci jsou odvozování ze stávající populace výpočtem ze tří náhodně vybraných jedinců podle diagramu na obrázku 15. Diagram představuje konkrétní funkci:

$$x_{o,g+1} = x_{2,g} + F \cdot \left(x_{3,g} - x_{Np-2,g}\right), \tag{4.35}$$

kde $x_{o,g+1}$ je nultý jedinec nové generace,

F je konstanta určující rychlost konvergence k výsledku,

$x_{2,g}$, $x_{3,g}$, $x_{Np-2,g}$ jsou tři náhodně vybraní jedinci původní generace g,

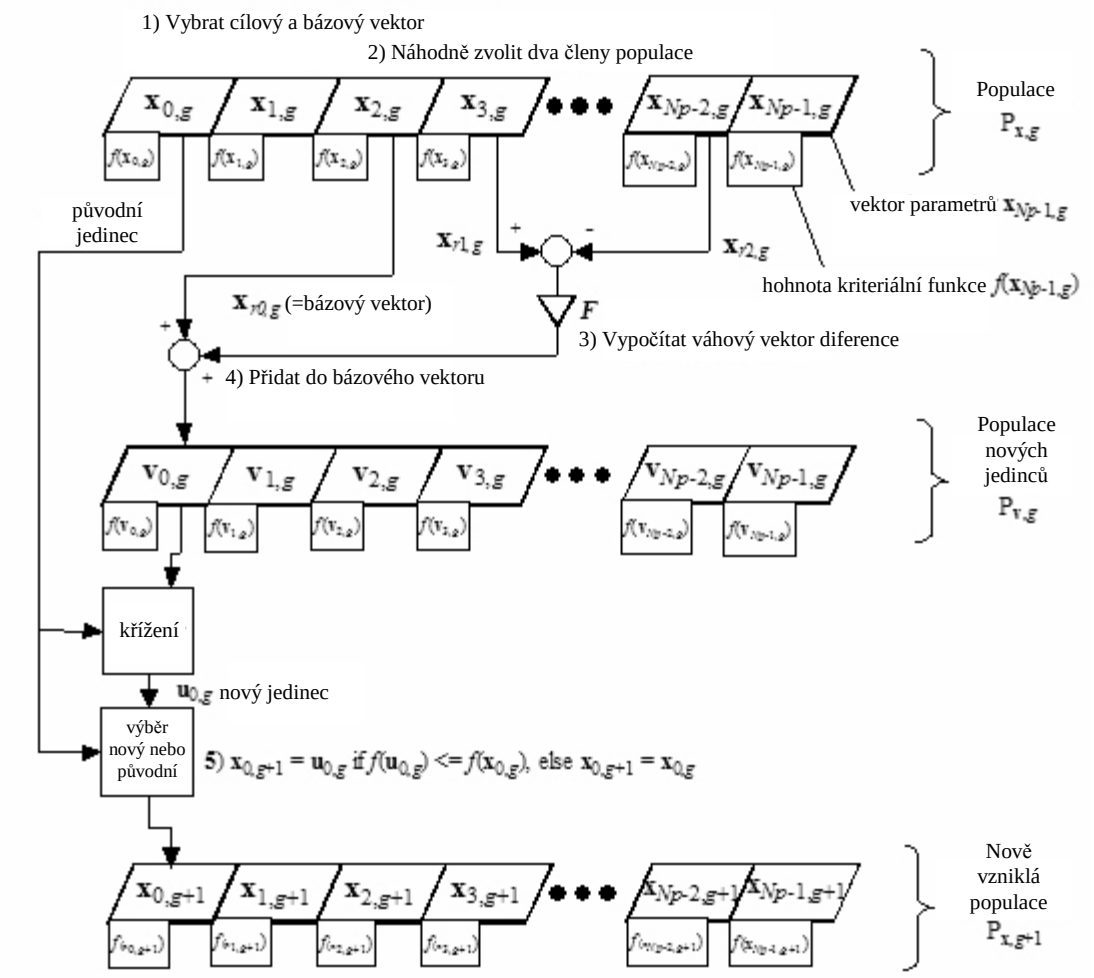
N_p je počet jedinců v generaci.

Platí, že čím bližší je hodnota F hodnotě 1, tím je rychlost konvergence vyšší, čím je menší, je algoritmus pomalejší, ale robustnější.

Obecně tedy:

$$x_{i,g+1} = x_{j,g} + F \cdot \left(x_{k,g} - x_{l,g}\right), \tag{4.36}$$

kde *i*-tý jedinec nové generace *g+1* vznikne kombinací *j*-tého, *k*-tého a *l*-tého, jedince původní generace *g*. Indexy *j, k, l* jsou pořadí jedince v rozsahu *0* až N_p*-1*.



Obrázek 14: Výpočet jednice nové populace pro diferenciální evoluční algoritmus [35]

Na počátku algoritmus vygeneruje N_p náhodných jedinců (možných řešení). Každé toto řešení dosadí do kriteriální funkce a ohodnotí jej (tzv. hodnota fitness). Čím nižší je hodnota fitness jedince, tím lepší představuje řešení (v případě hledání minima).

Další postup popíši v souladu s obrázkem 15. Vytvoříme *i*-tého jedince nové generace. Algoritmus náhodně vybere dva jedince. Rozdílem dvou jedinců vznikne diferenční směrový vektor, o který by se měl každý jedinec změnit, aby se přiblížil k výsledku. Tento vektor změny se přičte k jinému náhodně vybranému jedinci s vahou *F*. Síla této vazby ovlivní rychlost změny třetího vybraného jedince. Vypočtený jedinec může být zkřížen s jistou pravděpodobností, danou nastavením algoritmu, s *i*-tým jedincem původní generace. Nově vzniklý jedinec se ohodnotí fitness hodnotou. Je-li fitness nově vzniklého jedince lepší než *i*-tého jedince původní populace, je zařazen do nové populace. Není-li, je použit *i*-tý jedinec původní populace. Tento postup se opakuje do vytvoření požadovaného počtu jedinců.

Po celém výpočtu se ověřují podmínky ukončení výpočtu a v případě nesplnění těchto podmínek následuje výpočet nové generace.

4.5 Experimentální ověření

Algoritmus ladění vah pravidel diferenciální evolucí byl testován na dvou bázích znalostí. Na dvourozměrné funkci AND, která byla popsána úplným souborem vzorů a na modelové bázi znalostí identifikující poruchu automobilu, tak jak byla popsána ve [44]. Grafická reprezentace této báze znalostí je v příloze B Báze znalostí pro poruchu automobilu byla učena podle vzorů v tabulce 3.

Název vzoru	Vzor pro	Dotazy				Cílové uzly			Výsledná váha
		Starter	Radio	Nadrz	Kaluz	Bater	Palivo	Dira	
v01	Bater	95	50	x	x	9,1	x	x	90%
v02	Bater	0	100	x	x	9,1	x	x	
v03	Bater	0	0	x	x	90,5	x	x	
v04	Palivo	x	x	100	100	x	1,5	x	90% - nadrz 85% - kluz
v05	Palivo	x	x	100	0	x	60	x	
v06	Palivo	x	x	0	100	x	40	x	
v07	Palivo	x	x	0	0	x	98,5	x	
v08	Dira	x	x	100	100	x	x	83,3	80%
v09	Dira	x	x	100	0	x	x	16,7	
v10	Dira	x	x	25	25	x	x	31,8	
v11	Dira	x	x	0	100	x	x	16,7	

Tabulka 3: Pravděpodobnosti (v %) dotazovatelných a cílových uzlů ve vzorech

Algoritmus učení dává stabilní výsledky, tj. pro každou inicializaci evolučního algoritmu jsou výsledky shodné. Podmínka zastavení je nastavena tak, že když ve 100 po sobě jdoucích krocích nedojde ke zlepšení alespoň 0,01%, učení se ukončí. Algoritmus poskytoval výsledky v průměru za 380 ms. V tabulce 4 jsou srovnány dosažené výsledky. Porovnávány jsou hodnoty vah získané evolučním algoritmem s váhami, podle kterých byly v již nastavené bázi tvořeny vzory.

Cílový uzel	Alfa – požadované	Alfa - evoluč. alg.	Střední kvadratic. chyba přes všechny vzory (E)
Bater – vybitá baterie	90,0%	89.8%	8,15
Palivo – není palivo	90,0% | 85,0%	90% | 84,8%	10,87
Dira – palivo uniká	80,0%	80,0%	10,87

Tabulka 4: Srovnání vah získaných evolučním algoritmem s požadovanými hodnotami

4.6 Vícevrstvé báze znalostí

Ačkoliv se u praktických realizací bází znalostí programu NPS32 vyskytují převážně jednovrstvé struktury báze znalostí, tedy báze, kde dotazovatelné uzly jsou přes pravidlo navázány přímo na cílový uzel, mohou se obecně vyskytovat vícevrstvé báze znalostí s hlubší strukturou pravidel. Jednovrstvé (rozumějme vstupy, jedna vrstva pravidel, výstupy) báze znalostí programu NPS32 těží z agregačních vlastností cílového uzlu, tedy schopnosti skládat informace získané z pravidel. Naopak logické úlohy je poměrně obtížné takto realizovat. Matematický aparát expertního systému NPS32 umožňuje tvorbu libovolně strukturované báze, stejně jako vlastní program.

Při použití evolučních algoritmů pro optimalizaci komplexních vícevrstvých bází znalostí narážíme na problém velikosti stavového prostoru, kde není zaručeno nalezení nejlepšího řešení a nalezení vždy stejného řešení při opakované optimalizaci. Je tedy vhodnější využít některou z gradientních metod ladění vah pravidel v bázi znalostí. Pro optimalizaci vah pravidel gradientní metodou je ovšem nutné znát derivaci přenosové funkce pravidla.

Přenosová funkce pravidla s jedním antecedentem je dána výrazem

$$y_j^n = \frac{\left(\frac{w_j * F_{j1} - w_j + 1}{w_j * F_{j0} - w_j + 1}\right) \cdot F_{0y_j^n}}{\left(\frac{w_j * T_{j1} - w_j + 1}{w_j * T_{j0} - w_j + 1}\right) \cdot T_{0y_j^n} + \left(\frac{w_j * F_{j1} - w_j + 1}{w_j * F_{j0} - w_j + 1}\right) \cdot F_{0y_j^n}}, \tag{4.37}$$

kde w_j je vyjádření síly vazby *j*-tého pravidla,
(T_{j1}; F_{j1}) je pravděpodobnost *j*-tého pravidla po odpovědi dotazovatelných uzlů,
(T_{j0}; F_{j0}) je původní pravděpodobnost *j*-tého pravidla před odpovědí dotazovatelných uzlů,
$\left(T_{0y_j^n}; F_{0y_j^n}\right)$ je původní pravděpodobnost uzlu Y,
y_j^n je aktuální pravděpodobnost uzlu Y,
j je index příslušného pravidla.

Vyjádříme nyní gradient chybové funkce v závislosti na hodnotě příslušné váhy *w*. Podle pravidla o derivaci složené funkce přepíšeme vztah (4.34) na součin parciálních derivací

$$\frac{\partial \varepsilon_j^n}{\partial w_j^n} = \frac{\partial \varepsilon_j^n}{\partial y_j^n} \cdot \frac{\partial y_j^n}{\partial w_j^n}. \tag{4.38}$$

Zvolíme-li substituce:

$$A = \left(\frac{w_j * T_{j0} - w_j + 1}{w_j * T_{j0} - w_j + 1}\right) * T_{0y_j^n}, \tag{4.39}$$

$$B = \left(\frac{w_j * F_{j1} - w_j + 1}{w_j * F_{j0} - w_j + 1}\right) * F_{0y_j^n}, \tag{4.40}$$

$$C = \left(\frac{T_{j1} - 1}{w_j * T_{j0} - w_j + 1}\right) * T_{0y_j^n}, \tag{4.41}$$

$$D = \left(\frac{F_{j1} - 1}{w_j * F_{j0} - w_j + 1}\right) * F_{0y_j^n}, \tag{4.42}$$

$$E = \left(\frac{w_j * T_{j1} - w_j + 1}{\left(w_j * T_{j0} - w_j + 1\right)^2}\right) \cdot T_{0y_j^n}, \tag{4.43}$$

$$F = \left(\frac{w_j * F_{j1} - w_j + 1}{\left(w_j * F_{j0} - w_j + 1\right)^2} \right) \cdot F_{0 y_j^n}, \tag{4.44}$$

pak derivace $\frac{\partial y_j^n}{\partial w_j^n}$ je rovna

$$\frac{\partial y_j^n}{\partial w_j^n} = \frac{D - F * \left(F_{j1} - 1\right)}{A + B} - \frac{B}{\left(A + B\right)^2} \cdot \left(C - E \cdot \left(T_{j1} - 1\right) + D - F \cdot \left(F_{j1} - 1\right)\right), \tag{4.45}$$

kde pro výstupní vrstvu platí

$$\frac{\partial \varepsilon_j^{n_0}}{\partial y_j^{n_0}} = d_j - y_j^{n_0}. \tag{4.46}$$

Pro vnitřní vrstvy můžeme vyjádřit parciální derivaci $\frac{\partial \varepsilon_j^{n-1}}{\partial y_j^{n-1}}$, opět podle pravidla o derivování složené funkce, vztahem

$$\frac{\partial \varepsilon_j^{n-1}}{\partial y_j^{n-1}} = \sum_{y_j^{n-1} \to \Theta_j^n} \frac{\partial \varepsilon_j^n}{\partial y_j^n} \cdot \frac{\partial y_j^n}{\partial \alpha_j^n} \cdot \frac{\partial \alpha_j^n}{\partial y_j^{n-1}}, \tag{4.47}$$

kde Θ_j^{n+1} je *j*-té pravidlo v *n+1* vrstvě

$\frac{\partial \varepsilon_j^n}{\partial y_j^n}$ je chyba vypočtená pro nižší vrstvu pravidel, případně výstupní chyba pro nejnižší vrstvu pravidel,

α_j^n je vnitřní potenciál pravidla $\alpha_j^n = \frac{F_{j1}}{T_{j1} + F_{j1}}$.

Derivaci výstupu pravidla podle vnitřního potenciálu pravidla α_j^n, získáme ze vztahu (4.37) podle pravidla pro derivaci podílu:

$$\frac{\partial y_j^n}{\partial \alpha_j^n} = \frac{B'_\alpha \cdot \left(A + B\right) - B_\alpha \cdot \left(A'_\alpha + B'_\alpha\right)}{\left(A + B\right)^2}, \tag{4.48}$$

v tomto vztahu vyjádříme derivace vnitřních funkcí:

$$A'_\alpha = \left(\frac{w_j \cdot \left(T_{j1}\right)'_\alpha - w_j + 1}{w_j \cdot T_{j0} - w_j + 1} \right) \cdot T_{0 y_j^n}, \tag{4.49}$$

jelikož $\left(T_{j1}\right)'_\alpha$ závisí podle vztahu (4.1) na hodnotě F_{j1}, která je ovšem také závislá na hodnotě T_{j1}, nelze derivaci vyjádřit jak ukazují rovnice 4.50 a 4.51.

$$\left(T_{j1}\right)_{\alpha}' = \left(\frac{F_{j1}\cdot(1-\alpha)}{\alpha}\right)_{\alpha}' = \left(\frac{\frac{\alpha\cdot T_{j1}}{1-\alpha}\cdot(1-\alpha)}{\alpha}\right)_{\alpha}' \tag{4.50}$$

Úpravou rovnice (4.50) dostaneme identitu:

$$\left(T_{j1}\right)_{\alpha}' = \left(T_{j1}\right)_{\alpha}'. \tag{4.51}$$

Z výpočtu plyne, že matematický aparát expertního systému NPS32 nelze použít pro výpočet vah vícevrstvých grafů pravidel gradientními metodami. Pro výpočty vah jednovrstvých grafů je možné použít vztahy (4.38), (4.46) a vztahu (4.45) po dosazení substituovaných proměnných (4.39) až (4.44).

5 Analytická metoda přímého ladění vah pravidel v bázi znalostí

Kapitola popisuje algoritmus ladění vah pravidel v bázích znalostí expertních systémů. Je zde popsána matematická reprezentace báze znalostí a popis její struktury. Dále je zde odvozen gradientní algoritmus optimalizace vah pravidel báze znalostí na základě stanovené kriteriální funkce. Optimalizační algoritmus byl experimentálně ověřen odladěním vah pravidel třech reálných bází znalostí popsaných v kapitole 6.

Algoritmus přímého ladění vah pravidel v bázi znalostí byl implementován do nového expertního systému RESLA, který byl vytvořen pro testování tohoto algoritmu.

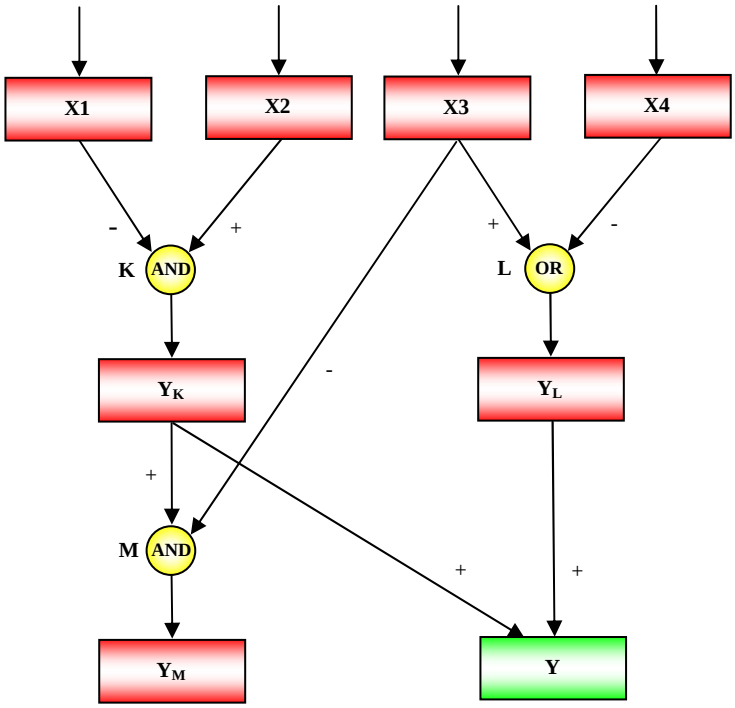
5.1 Báze znalostí expertního systému RESLA

Znalostní báze expertního systému RESLA využívajícího metodu přímého nastavování vah pomocí modifikovaného algoritmu back-propagation tvoří orientovaný acyklický graf pravidel. Jednoduchý příklad je uveden na obrázku 15.

Uzly v první vrstvě reprezentují vstupní hypotézy, tedy hodnoty odpovědí na dotazy položené uživateli. Uzly vnitřních vrstev reprezentují podpůrné hypotézy, tedy takové, ze kterých se odvozuje výsledná hypotéza (doporučení) v poslední vrstvě. Uzly výstupní vrstvy představují závěr předkládaný uživateli korespondující s jeho odpověďmi (obrázek 15, červeně – logický uzel, zeleně – agregační pravidlo).

Báze znalostí je složena z pravidel realizující funkce AND a OR libovolného počtu vstupních proměnných (nejde tedy pouze o binární operaci, jak je u těchto funkcí obvyklé) a funkci pro kompozici výsledků. Negace v konjunktivních a disjunktivních pravidlech je realizována doplňkem vstupní proměnné, ve funkci kompozice jsou negované vstupní proměnné násobeny -1. Lze tedy realizovat bázi znalostí libovolné složitosti a velikosti, neboť trojce funkcí AND, OR a NEGACE tvoří úplný systém logických funkcí, což znamená, že kombinací těchto funkcí lze realizovat jakoukoli jinou logickou funkci.

Pro správnou funkci expertního systému, tedy sytému, který řadí výsledné hypotézy podle pravděpodobnosti nebo jiného faktoru neurčitosti, jsou tato logická pravidla navázána na agregační výstupní pravidlo (rovnice 5.10, obrázek 15 - zeleně).



Obrázek 15: Příklad struktury báze znalostí expertního systému používané algoritmem přímého učení

Podmínkou použití pravidel v učicím algoritmu, jak bude patrné dále v kapitole 5.2, je diferencovatelnost přenosové funkce pravidla. Přenosová funkce pravidla (rovnice 5.1) může obsahovat negaci některé nebo všech vstupních proměnných, obsahuje funkci (realizující AND nebo OR) všech vstupních proměnných a násobení váhou *w*.

$$y_j^n = w_j^n \cdot \Theta(\mathbf{x}), \tag{5.1}$$

kde y_j^n je výstup *j*-tého pravidla v *n*-té vrstvě,

w_j^n je váha *j*-tého pravidla v *n*-té vrstvě,

$\Theta(\mathbf{x})$ je funkce reprezentující funkci AND nebo OR pravidla,

$\boldsymbol{x}$ je vektor vstupů pravidla.

Z rovnice (5.1) je patrné, že podmínka diferencovatelnosti přenosové funkce pravidla bude splněna, bude-li diferencovatelná funkce $\Theta(\mathbf{x})$.

Standardní binární logické funkce AND a OR diferencovatelné nejsou, což byl i problém algoritmu sémantické konverze na neuronovou síť(popsanou v kapitole 2.3.1). Byla proto zvolena náhrada těchto funkcí funkcemi t-normou (s-konormou) pro logickou funkci AND a s-normou pro logickou funkci OR v pravděpodobnostní formě [22].

Jsou definovány:

pro t-normu

$$y = \prod_{i}^{N} x_i \,, \tag{5.2}$$

a pro s-normu

$$y = x_1 + x_2 - x_1 \cdot x_2 \,, \tag{5.3}$$

kde y je výstup funkce,
x_i jsou vstupní proměnné,
i je počet vstupních proměnných.

Jako vnitřní funkci pravidla můžeme použít jakoukoli vhodnou hladkou funkci. Například z hlediska informační kapacity báze by bylo vhodné použít rovnice (4.8) uvedené v kapitole 4.1, což ovšem nelze, jak bylo dokázáno v kapitole 4.6.

Pravidla použitá pro bázi znalostí jsou definována:

konjunktivní pravidlo

$$y_j^n = w_j^n \cdot \prod_{i=1}^{N} \left(\lambda_j^n \left(y_i^{n-1} \right) \right), \tag{5.4}$$

kde y_j^n je výstup j-tého pravidla v n-té vrstvě báze,

w_j^n je váha j-tého pravidla v n-té vrstvě báze,

y_i^{n-1} je výstup i-tého pravidla v *n-1* vrstvě báze, tedy pravidla vstupujícího jako argument do j-tého pravidla,

N je počet vstupů j-tého pravidla.

funkce $\lambda_j^n(x)$ je definována:

$$\lambda_j^n = x \,, \text{ pro pozitivní vazbu,} \tag{5.5}$$

$$\lambda_j^n = 1 - x \,, \text{ pro negativní vazbu,} \tag{5.6}$$

disjunktní pravidlo

$$y_j^n = w_j^n \cdot \underset{i=1}{\overset{N}{OR}} \left(\lambda_j^n \left(y_i^{n-1} \right) \right), \tag{5.7}$$

kde y_j^n je výstup j-tého pravidla v n-té vrstvě báze,

w_j^n je váha j-tého pravidla v n-té vrstvě báze,

y_i^{n-1} je výstup i-tého pravidla v *n-1* vrstvě báze, tedy pravidla vstupujícího jako argument do j-tého pravidla,

N je počet vstupů j-tého pravidla.

Funkce $\underset{i=1}{\overset{N}{OR}}(\mathbf{x})$ je definována jako rekurzivní funkce:

$$\underset{i=1}{\overset{N}{OR}}(\mathbf{x}) = \bigcup(x_1, \bigcup(x_2, ... \bigcup(x_{i-1}, x_i))), \tag{5.8}$$

kde

$$\bigcup(x_i, x_j) = x_i + x_j - x_i \cdot x_j \tag{5.9}$$

agregační pravidlo (funkce)

$$y_j^n = w_j^n \cdot \sum_{i=1}^{N} \left(\gamma_i^n \left(y_i^{n-1}\right)\right), \tag{5.10}$$

kde y_j^n je výstup *j*-tého pravidla v *n*-té, poslední, vrstvě báze,

y_i^{n-1} je výstup *i*-tého pravidla v *n-1* vrstvě báze, tedy pravidla vstupujícího jako argument do *j*-tého pravidla,

N je počet vstupů *j*-tého pravidla,

funkce $\gamma_j^n(x)$ je definována

$$\gamma_j^n = x \text{, pro pozitivní vazbu,} \tag{5.11}$$

$$\gamma_j^n = -x \text{, pro negativní vazbu.} \tag{5.12}$$

Báze znalostí je tedy tvořena pravidly s přenosovou funkcí tvořenou t-normou nebo s-normou využívajícími klasický pravděpodobnostní koncept k zpracování neurčitosti. Agregační funkce zajišťuje skládání přínosů jednotlivých podstromů báze do výsledné pravděpodobnosti cílového uzlu (hypotézy). Kombinací agregačního pravidla s ostatními logickými pravidly lze získat systém, který na základě informace ze vstupního pravidla zvyšuje nebo snižuje pravděpodobnost cílového uzlu.

5.2 Metoda nastavování vah pravidel v bázi znalostí

Metoda automatického nastavování vah je založena na modifikaci algoritmu back-propagation [42]. Jde o metodu učení s učitelem implementující delta pravidlo, což je pravidlo využívající k úpravě vah (resp. váhy) gradient chybové funkce. K činnosti algoritmu je nutné znát vzory, tedy několik různých příkladů odezvy systému na vstupní informaci.

Obdobně jako v algoritmu back-propagation se vypočítává chyba sítě při předložení vzorů a adaptují se váhy jednotlivých pravidel na základě zpětně šířené odchylky vzoru od výstupu. Metoda nastavování vah ovšem využívá jiný model pravidla, kterému se musel přizpůsobit i matematický aparát.

Na začátku adaptace jsou váhy ***w*** pravidel nastaveny náhodně v intervalu <0,1>. V každém kroku učení *k=1,2,3...* je bázi předložen jeden vzor z trénovací množiny. Algoritmus se snaží nastavit váhy pravidel v bázi tak, aby odpovídaly předloženému

vzoru. Adaptace vah probíhá v tzv. tréninkových cyklech, ve kterých se postup opakuje pro každý tréninkový vzor.

Jako první je třeba vytvořit učební vzory pro všechny vstupy a výstupy. Mějme tedy množinu vzorů

$$\mathbf{M} = \{\mathbf{d}, \mathbf{y}\}, \tag{5.13}$$

kde **d** je vektor vstupů,
y je vektor výstupů,
M je množina vzorů.

Znalostní báze jako celek představuje projekci vstupů na výstupy

$$\Psi: \quad \mathbf{x} \rightarrow \mathbf{y}, \tag{5.14}$$

kde Ψ je funkce reprezentující projekci vstupního vektoru na výstupní.

Přenosová funkce sítě je

$$\mathbf{y} = \Psi(\mathbf{x}), \tag{5.15}$$

kde **y** je vektor výstupů sítě,
x je vektor vstupů sítě,
Ψ přenosová funkce sítě

Chyba sítě je potom definována

$$\mathbf{e} = \mathbf{d} - \mathbf{y} = \mathbf{d} - \Psi(\mathbf{x}), \tag{5.16}$$

kde **d** je požadovaná hodnota výstupu.

Pro jeden výstup logického pravidla plyne z rovince 5.1 dosazením do 5.16 okamžitá kvadratická chyba

$$e_j = \frac{1}{2}\sum_{p=1}^{M}\left({}_p d_j - {}_p w_j^n \cdot {}_p y_{j\left(\mathbf{w}_j, {}_p\mathbf{y}_j^{n-1}, \lambda_j\right)}\right)^2 = \frac{1}{2}\sum_{p=1}^{M}\left({}_p d_j - {}_p w_j^n \cdot {}_p \Theta_j\left(\lambda_j^n\left({}_p y_j^{n-1}\right)\right)\right)^2, \tag{5.17}$$

kde M je počet příslušných vzorů j-tému výstupu,

p je příslušný vzor,

pro jeden výstup agregačního pravidla plyne z rovnice 5.10 dosazením do 5.16 okamžitá kvadratická chyba

$$e_j = \frac{1}{2}\sum_{p=1}^{M}\left({}_p d_j - {}_p w_j^n \cdot {}_p y_{j\left(\mathbf{w}_j, {}_p\mathbf{y}_j^{n-1}, \gamma_j\right)}\right)^2 = \frac{1}{2}\sum_{p=1}^{M}\left({}_p d_j - {}_p w_j^n \cdot \sum_{i=1}^{N}\left(\gamma_j^n\left(y_i^{n-1}\right)\right)\right)^2, \tag{5.18}$$

kde M je počet příslušných vzorů j-tému výstupu,

p je příslušný vzor.

Kritériem kvality je tedy chybová funkce:

$$\varepsilon = \mathrm{E}\{e_j\}. \tag{5.19}$$

Váhy sítě jsou nastavovány pomocí metody nestrmějšího sestupu delta pravidlem

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu_k \nabla e, \tag{5.20}$$

kde μ_k je učící konstanta udávající rychlost učení,

k je krok učení.

Rozepsáním tohoto vztahu do složek dostáváme výraz

$$w_j^n(k+1) = w_j^n(k) - \mu_k \frac{\partial e}{\partial w_j^n}(k), \tag{5.21}$$

pro $n = n_0$, kde n_0 je počet vrstev (index výstupní vrstvy).

Přenosová funkce pravidla je podle 5.1 dána výrazem

$$y_j^n = w_j^n \cdot \alpha_j^n = w_j^n \cdot \Theta_j\left(\lambda_j^n\left(y_j^{n-1}\right)\right), \tag{5.22}$$

kde α_j^n je vnitřní potenciál pravidla, tedy hodnota funkce Θ(.),

Θ(.) reprezentuje funkci AND pro konjunktivní pravidlo a funkci OR pro disjunktní pravidlo.

V následující části bude vyjádříme gradient chybové funkce 5.16 v závislosti na hodnotě příslušné váhy *w*. Podle pravidla o derivaci složené funkce bude vztah 5.17 (chybová funkce pro logická pravidla) přepsán na součin parciálních derivací

$$\frac{\partial \varepsilon_j^n}{\partial w_j^n} = \frac{\partial \varepsilon_j^n}{\partial y_j^n} \cdot \frac{\partial y_j^n}{\partial w_j^n} = \frac{\partial \varepsilon_j^n}{\partial y_j^n} \cdot \alpha_j^n, \tag{5.23}$$

kde $\alpha_j^{n_0}$ je vnitřní potenciál pravidla.

Pro výstupní vrstvu platí z rovnice 5.17 pro logická pravidla

$$\frac{\partial \varepsilon_j^n}{\partial w_j^n} = \frac{1}{2} 2\left(d_j - w_j \cdot \Theta\left(\mathbf{x}_j\right)\right)\left(-\Theta\left(\mathbf{x}_j\right)\right) = -\left(d_j - y_j^{n_0}\right) \cdot \alpha_j^{n_0}, \tag{5.24}$$

kde $\mathbf{x}_j$ reprezentuje vstupy do jádra funkce pravidla, tedy funkci $\lambda_j^n\left(y_j^{n-1}\right)$,

$\alpha_j^{n_0}$ je vnitřní potenciál výstupního pravidla,

pro agregační pravidlo platí z rovnice 5.18

$$\frac{\partial \varepsilon_j^n}{\partial w_j^n} = \frac{1}{2} 2\left(d_j - w_j \cdot \sum_{i=1}^{N}\left(\gamma_i^n\left(y_i^{n-1}\right)\right)\right)\left(-\sum_{i=1}^{N}\left(\gamma_i^n\left(y_i^{n-1}\right)\right)\right) = -\left(d_j - y_j^{n_0}\right) \cdot \alpha_j^{n_0}, \tag{5.25}$$

kde $\mathbf{x}_j$ reprezentuje vstupy do jádra funkce pravidla, tedy funkci $\lambda_j^n\left(y_j^{n-1}\right)$,

$\alpha_j^{n_0}$ je vnitřní potenciál výstupního pravidla,

tedy

$$\frac{\partial \varepsilon_j^{n_0}}{\partial y_j^{n_0}} = y_j^{n_0} - d_j. \tag{5.26}$$

Pro vnitřní vrstvy můžeme být vyjádřena parciální derivaci vztahem

$$\frac{\partial \varepsilon_j^{n-1}}{\partial y_j^{n-1}} = \sum_{y_j^{n-1} \to \Theta_j^n} \frac{\partial \varepsilon_j^n}{\partial y_j^n} \cdot \frac{\partial y_j^n}{\partial \alpha_j^n} \cdot \frac{\partial \alpha_j^n}{\partial y_j^{n-1}} = \sum_{y_j^{n-1} \to \Theta_j^n} \frac{\partial \varepsilon_j^n}{\partial y_j^n} \cdot w_j^n \cdot \frac{\partial \Theta_j^n\left(\lambda_j^n\left(y_j^{n-1}\right)\right)}{\partial y_j^{n-1}}, \tag{5.27}$$

kde Θ_j^{n+1} je *j*-té pravidlo v *n+1* vrstvě.

Agregační pravidlo se ve vnitřních vrstvách nevyskytuje.

Váhy výstupní *n*-té vrstvy jsou opravovány podle vztahu

$$w_j^{n_0}(t+1) = w_j^{n_0}(t) + \mu_t \alpha_j^{n_0}\left(y_j^{n_0} - d_j^{n_0}\right), \tag{5.28}$$

váhy vnitřních vrstev podle vztahu

$$w_j^{n-1}(t+1) = w_j^{n-1}(t) + \mu_t \alpha_j^{n-1} \sum_{y_j^{n-1} \to \Theta_j^n} \frac{\partial \varepsilon_j^n}{\partial y_j^n} \cdot w_j^n \cdot \frac{\partial \Theta_j^n\left(\lambda_j^n\left(y_j^{n-1}\right)\right)}{\partial y_j^{n-1}}, \tag{5.29}$$

kde $n \le n_0$.

Výraz $\frac{\partial \Theta_j^n\left(\lambda_j^n\left(y_j^{n-1}\right)\right)}{\partial y_j^{n-1}}$ ze vztahu (5.17) je možné rozepsat pro konkrétní případy takto:

pro konjunktivní pravidlo s pozitivní vazbou derivované vstupní proměnné

$$\frac{\partial \Theta_j^n\left(\lambda_j^n\left(y_j^{n-1}\right)\right)}{\partial y_j^{n-1}} = \prod_{\substack{i=0 \\ i \ne j}}^{N} \left(\lambda_j^n\left(y_j^{n-1}\right)\right), \tag{5.30}$$

pro konjunktivní pravidlo s negativní vazbou derivované vstupní proměnné

$$\frac{\partial \Theta_j^n\left(\lambda_j^n\left(y_j^{n-1}\right)\right)}{\partial y_j^{n-1}} = -\prod_{\substack{i=0 \\ i \ne j}}^{N} \left(\lambda_j^n\left(y_j^{n-1}\right)\right), \tag{5.31}$$

pro disjunktivní pravidlo s pozitivní vazbou derivované vstupní proměnné

$$\frac{\partial \Theta_j^n\left(\lambda_j^n\left(y_j^{n-1}\right)\right)}{\partial y_j^{n-1}} = 1 - \bigcup_{\substack{i=0 \\ i \ne j}}^{N} \left(\lambda_i^n\left(y_i^{n-1}\right)\right), \tag{5.32}$$

pro disjunktivní pravidlo s negativní vazbou derivované vstupní proměnné

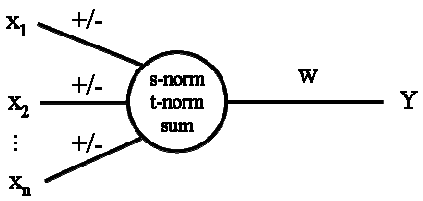
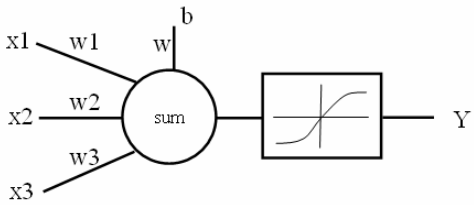
$$\frac{\partial \Theta_j^n\left(\lambda_j^n\left(y_j^{n-1}\right)\right)}{\partial y_j^{n-1}} = \bigcup_{\substack{i=0 \\ i \ne j}}^{N} \left(\lambda_i^n\left(y_i^{n-1}\right)\right) - 1. \tag{5.33}$$

V každém kroku učení je stanovena chyba všech výstupů sítě pravidel použitím rovnice (5.16), a s ohledem na potenciál pravidla (hodnota funkce Θ(.)), jsou adaptovány váhy výstupní vrstvy sítě dle rovnice (5.28). Chyby následně šíříme zpět grafem použitím rovnice (5.27). Váhy ve skrytých vrstvách jsou adaptovány podle rovnice (5.29), opět s ohledem na potenciál pravidla a váhy a chyby z pravidel z nižších vrstev spojených s tímto pravidlem.

Proces učení je následující:

1. Inicializace vah v síti prioritními váhami (pokud existují), jinak náhodně v rozmezí 0-1.
2. Nastavit vzor na vstupy sítě.
3. Přepočítat síť.
4. Vypočíst chybu každého výstupu na základě vzoru dle rovnice (5.16).
5. Vypočíst celkovou chybu sítě podle rovnice (5.19).
6. Adaptovat váhy výstupní vrstvy podle rovnice (5.28).
7. Adaptovat váhy ve skrytých vrstvách podle rovnice (5.29).
8. Opakovat kroky 2 – 7 pro všechny vzory.
9. Jestliže je celková chyba sítě menší než požadovaná, nebo bylo dosaženo maximálního počtu kroků, tak skončit, jinak pokračuj bodem 2.

Následující tabulka ukazuje hlavní odlišnosti modelu a algoritmu učení pro pravidlovou bázi znalostí a neuronovou síť.

	pravidlová síť	**neuronová síť**
Model		
Přenosová funkce	s-norma, t-norma, suma	sigmoida, lineární, omezená ...
Topologie	případově propojená síť	plně propojená síť
Šíření chyby	$\sum_{y_j^{n-1}\to\Theta_j^n} \frac{\partial \varepsilon_j^n}{\partial y_j^n} \cdot w_j^n \cdot \frac{\partial \Theta_j^n\left(\lambda_j^n\left(y_j^{n-1}\right)\right)}{\partial y_j^{n-1}}$	$\sum_{r\in j^{\rightarrow}} \frac{\partial E_k}{\partial y_r} \cdot \frac{\partial y_r}{\partial \alpha_r} \cdot \frac{\partial \alpha_r}{\partial y_j} =$ $\sum_{r\in j^{\rightarrow}} \frac{\partial E_k}{\partial y_r} \cdot \lambda_r y_r (1-y_r) w_{rj}$

Tabulka 5 Rozdíly modelů a algoritmů učení pravidlových a neuronových sítí

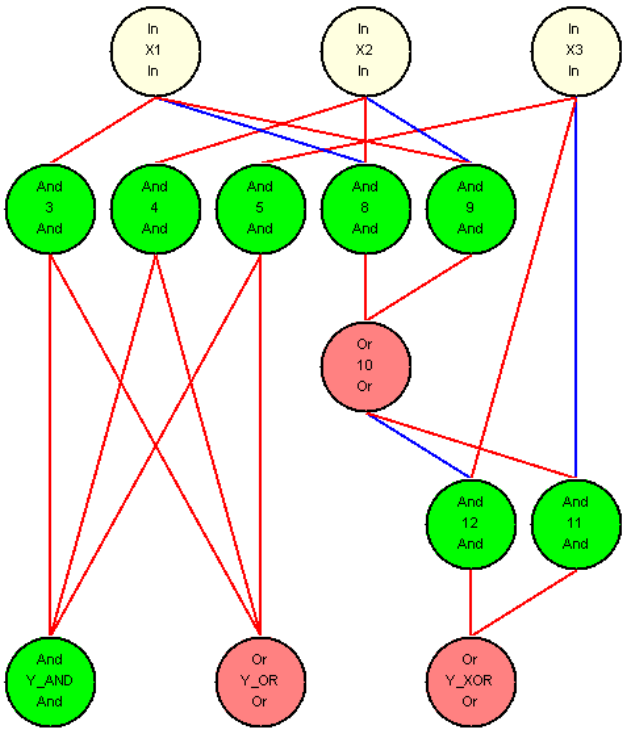
6 Experimentální ověření

Kapitola popisuje experimentální ověření analytického algoritmu přímého ladění vah pravidel v bázi znalostí expertního systému RESLA z kapitoly 5. Jsou zde popsány tři reálné báze znalostí, jejich struktura, tvorba trénovacích i testovacích vzorů a vyhodnocení procesu adaptace vah pravidel.

Algoritmus přímého nastavování vah v bázi znalostí byl testován na třech rozdílných úlohách. První z úloh je báze znalostí realizující logické funkce AND, OR a XOR pro ověření funkčnosti a správnosti algoritmu. Druhou úlohou je poměrně velká a komplexní báze znalostí z reálného prostředí, na které bylo testováno chování algoritmu na složitých úlohách. Třetí úlohou bylo ověření proti existující odladěné bázi znalostí, se kterou byl algoritmus porovnán co do účinnosti a schopnosti generalizace.

6.1 Logická úloha

Úloha představuje pokus na úrovni jednoduché logické struktuy. Jejím cílem je ověřit funkčnost a správnost algoritmu. Úlohou je natrénovat bázi pravidel tak, aby odpovídala svými výsledky předem dané bázi, podle které byly pro trénovanou bázi tvořeny vzory.



Obrázek 16: Grafická podoba báze znalostí pro logickou úlohu (z expertního systému RESLA)

Pravidla v bázi znalostí jsou definována podle rovnic 6.1, které představují strukturu báze znalostí z obrázku 16. Číslo v závorce za rovnicí je hodnota

pravděpodobnosti, kterou by měla síť dosáhnout po naučení, resp. hodnota váhy pravidla v bázi, podle které byly tvořeny vzory.

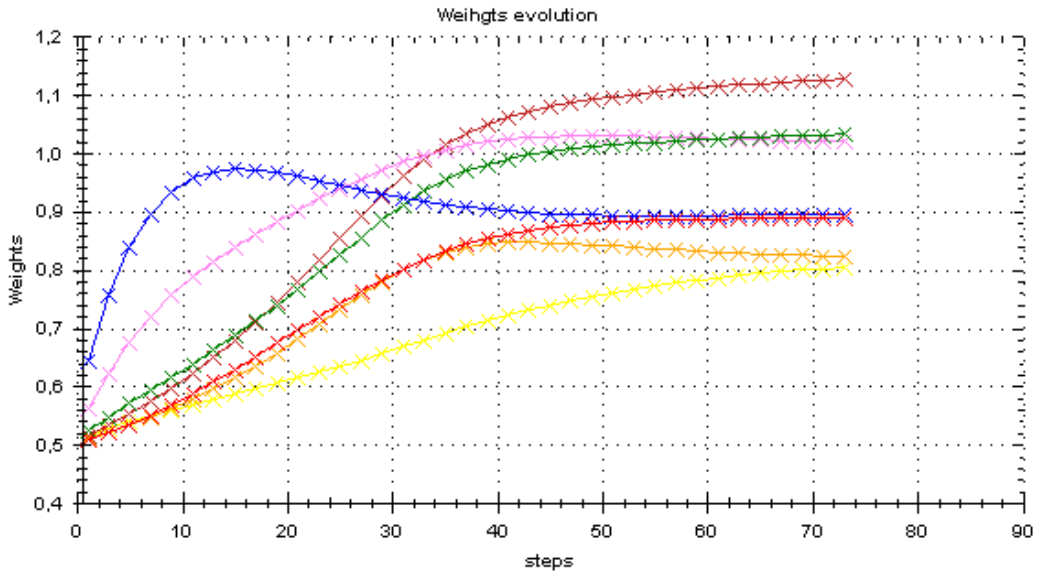
$$\begin{array}{llll} X_1 \wedge X_2 \wedge X_3 \rightarrow Y_{AND} & (0{,}8) & H_1 \vee H_2 \rightarrow H_3 & (1{,}0) \\ X_1 \vee X_2 \vee X_3 \rightarrow Y_{OR} & (0{,}9) & \neg H_3 \wedge X_3 \rightarrow H_4 & (1{,}0) \\ \neg X_1 \wedge X_2 \rightarrow H_1 & (1{,}0), & H_3 \wedge \neg X_3 \rightarrow H_5 & (1{,}0) \\ X_1 \wedge \neg X_2 \rightarrow H_2 & (1{,}0) & H_4 \vee H_5 \rightarrow Y_{XOR} & (0{,}85) \end{array} . \qquad (6.1)$$

Báze znalostí byla trénována následujícími učebními vzory, kde první množina reprezentuje vstupy báze znalostí, druhá množiny požadované výstupy:

$$\begin{array}{l} \{1,1,1\} \rightarrow \{0.8, 0.9, 0.85\} \\ \{0,1,1\} \rightarrow \{0.0, 0.9, 0.0\} \\ \{1,0,1\} \rightarrow \{0.0, 0.9, 0.0\} \\ \{1,0,0\} \rightarrow \{0.0, 0.9, 0.85\} \end{array} .$$

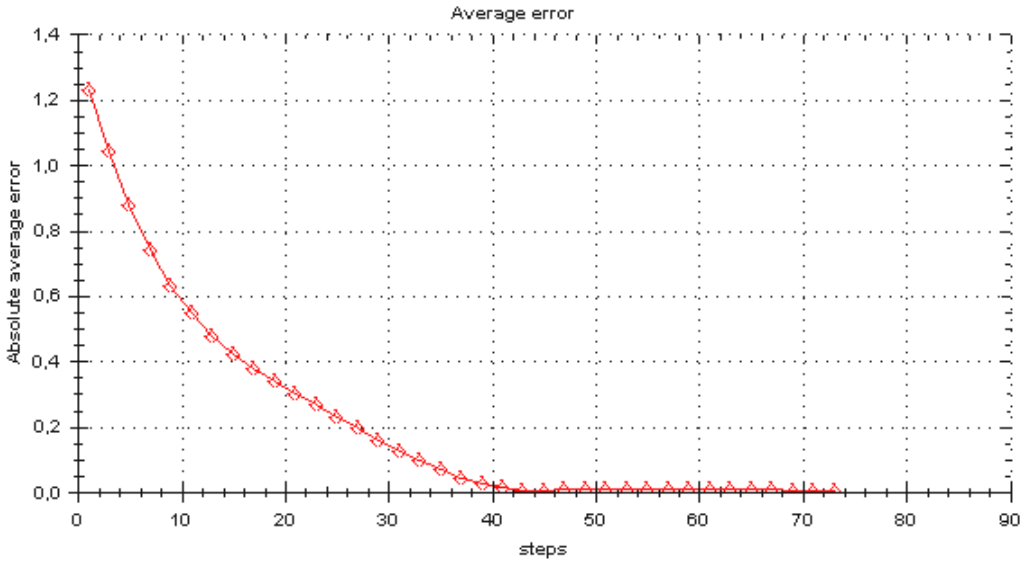
Algoritmus přímého nastavování vah pravidel v bázi znalostí iterativně upravuje iniciální váhy pravidel (náhodné nebo předem dané) tak, aby výstupy báze znalostí správně klasifikovaly trénovací vzory. V následujících grafech je patrný vývoj vah a průměrné globální chyby klasifikace báze pravidel v průběhu učení.

Graf na obrázku 17 představuje vývoj vybraných vah pravidel v bázi znalostí pro koeficient učení 0,1. Graf na obrázku 18 představuje vývoj průměrné absolutní chyby klasifikace báze znalostí v průběhu učení pro tentýž koeficient učení (0,1). Průměrná absolutní chyba po naučení báze znalostí byla menší než 0,01 [2].



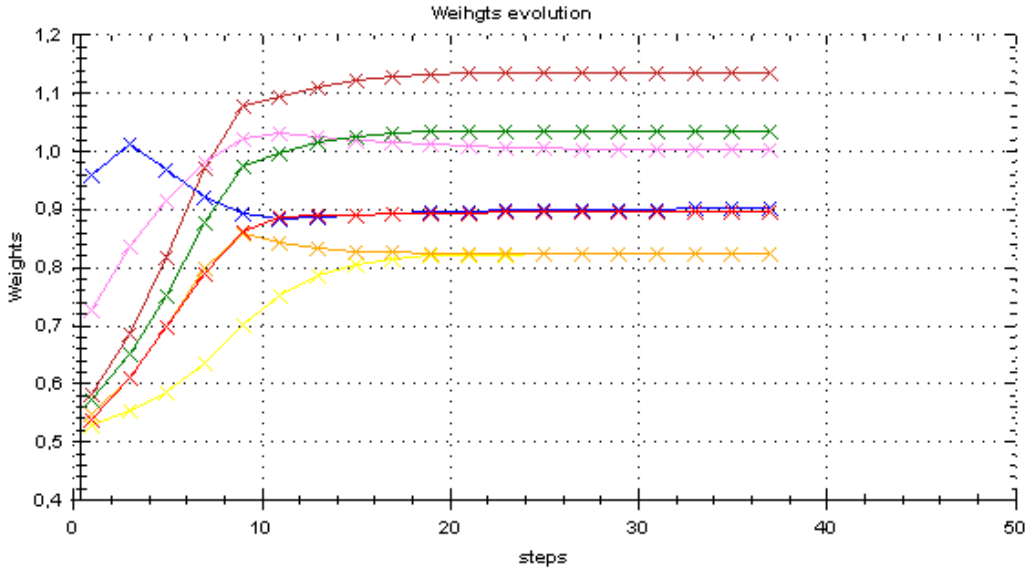
Obrázek 17: Vývoj vah pravidel pro testovanou logickou úlohu a koeficient učení: 0.1

[2] Hodnoty chybové funkce jsou v expertním systému RESLA zaokrouhlovány na setiny (jednotky procenta), proto je použito výrazu „je menší než 0,01“, i když program ukazuje hodnotu 0,00, neboť vlivem zaokrouhlování se neuplatní hodnoty chybové funkce menší než 0,005.

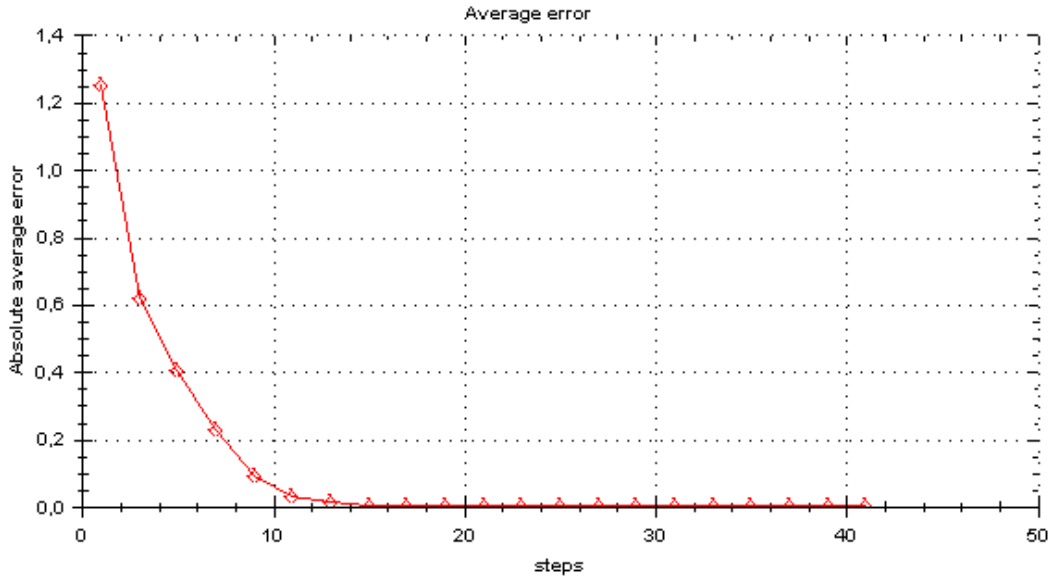


Obrázek 18: Vývoj globální chyby pro testovanou logickou úlohu a koeficient učení: 0.1

Grafy na obrázcích 19 a 20 představují vývoj vah a průměrné absolutní chyby klasifikace báze znalostí pro koeficient učení 0,4. Průměrná absolutní chyba po naučení báze znalostí byla menší než 0,01.



Obrázek 19: Vývoj vah pravidel pro testovanou logickou úlohu a koeficient učení: 0.4



Obrázek 20: Vývoj globální chyby pro testovanou logickou úlohu a koeficient učení: 0.4

Báze znalostí byla testována na úplný soubor vzorů podle tabulky 6.

X1	X2	X3	AND	OR	XOR
0	**0**	**0**	0	0	0
0	**0**	**1**	0	0,9	0,85
0	**1**	**0**	0	0,9	0,85
0	**1**	**1**	0	0,9	0
1	**0**	**0**	0	0,9	0,85
1	**0**	**1**	0	0,9	0
1	**1**	**0**	0	0,9	0
1	**1**	**1**	0,8	0,9	0,85

Tabulka 6: Testovací vzory pro logickou bázi znalostí

Jak je patrné z grafů, průměrná chyba klasifikace báze znalostí pro logickou úlohu byla menší než 1 % pro trénovací vzory, pro obě nastavení koeficientu učení. Chyba klasifikace báze znalostí pro testovací vzory z tabulky 6 byla také menší než 1 %. Detailnější výsledky, jako výstupy, chyby výstupu a průměrnou chybu báze pro každý vzor, je možné vidět přímo v programu RESLA. Z grafů na obrázcích 17 až 20 je vidět vliv koeficientu učení na rychlost adaptace vah a rychlost poklesu globální chyby báze znalostí. Pro vyšší hodnoty koeficientu učení je adaptace vah pravidel rychlejší, což je způsobeno větším vlivem chybové hodnoty (rozdíl výstupní a požadované hodnoty báze) šířené zpětně bází pravidel. Pro komplexní báze znalostí byla vhodná hodnota koeficientu učení experimentálně stanovena na 0,1. U komplexních bází znalostí může vlivem vysoké hodnoty koeficientu učení dojít k rozkmitání učicího algoritmu.

6.2 Báze znalostí pro diagnostiku poruch srdečního rytmu

Báze znalostí se věnuje problematice diagnostiky bradyarytmií srdečního rytmu na základě parametrů signálu EKG. Cílem této úlohy je ověřit chování učícího algoritmu na komplexní reálné bázi znalostí a ověřit tak výhody uplatnění algoritmu v praxi.

6.2.1 Stručná teorie k bázi znalostí pro diagnostiku bradyarytmií

Poruchy srdečního rytmu nazývané odborně **arytmie** patří mezi nejčastější srdeční onemocnění. Vznikají jako důsledek odlišného vytváření nebo vedení elektrických vzruchů v srdci. Ve většině případů jde o naprosto nezávažné arytmie, které si postižený člověk vůbec neuvědomuje, a které lze zachytit pouze dlouhodobým monitorováním elektrokardiogramu (EKG). Kromě toho existuje celá řada záchvatovitých nebo setrvalých poruch srdečního rytmu, ať ve smyslu plus (rytmus rychlejší než normálně) nebo minus (pomalý rytmus nebo dlouhé pauzy v srdeční činnosti), které mohou působit nemocnému celou řadu obtíží. Zatímco u jinak zdravých lidí nepředstavují tyto arytmie až na výjimky bezprostřední ohrožení života, u nemocných s postižením srdce (například po infarktu myokardu) mohou být některé arytmie životu nebezpečné. [citováno z: [14]]

Typy arytmií

Srdce se normálně stahuje asi 60-100-krát za minutu, přičemž elektrické podráždění síní předchází aktivaci komor. Tento normální rytmus se nazývá sinusovým rytmem. Někdy i za normálních okolností pracuje srdce pomaleji (například ve spánku) nebo rychleji (například při cvičení). Při srdečních arytmiích bývá rytmus srdce abnormálně pomalý (bradyarytmie) nebo naopak rychlý (tachyarytmie). V prvním případě se buď elektrický vzruch v sinusovém uzlu tvoří pomalu nebo je porušeno jeho vedení přes síňokomorový uzel do komor. Ve druhém případě se buď stane místem tvorby rychlých elektrických vzruchů kterákoliv jiná malá oblast svaloviny síní nebo komor nebo elektrický impulz krouží v různě veliké oblasti srdce kolem dokola a aktivuje okolní svalovinu. Pochází-li rychlý rytmus ze svaloviny síní nebo oblasti síňokomorového uzlu, nazývá se arytmie supraventrikulární. Naopak, pochází-li porucha srdečního rytmu ze svaloviny komor, je označována jako komorová. [citováno z: [14]]

Bradyarytmie

Příznaky pomalého srdečního rytmu (bradyarytmie) se mohou lišit podle toho o jakou poruchu jde a jak rychle vzniká. V případech, kdy je porušena normální tvorba elektrických vzruchů v sinusovém uzlu, pracuje srdce pomalu a nedokáže zvýšit svoji činnost při zátěži. Tehdy trpí nemocní závratěmi, točením hlavy nebo zvýšeným zadýcháváním a únavností při zátěži. Pokud je srdeční akce velmi pomalá nebo jsou přítomny několikavteřinové výpady v tvorbě elektrických vzruchů, může dojít i ke krátkodobé ztrátě vědomí. Náhlá porucha vedení elektrických vzruchů ze síní na komory se projeví obvykle jako krátkodobá ztráta vědomí. Vznikne-li porucha postupně nebo je-li přechodného charakteru, mohou být příznaky podobné jako u poruchy tvorby elektrických vzruchů v sinusovém uzlu (točení hlavy, únavnost aj.). Je však nutno zdůraznit, že podobné příznaky mohou být způsobeny celou řadou dalších onemocnění (např. epilepsie, mozkové příhody apod.), a proto je potřeba odborného posouzení celé situace lékařem. [citováno z: [14]]

6.2.2 Struktura báze znalostí pro diagnostiku bradyarytmií

Báze znalostí je tvořena osmi dotazovatelnými vstupními uzly, které zastupují tyto parametry signálu EKG:

- Délka intervalu PR
- Charakteristika intervalu PR
- Pravidelnost komplexu QRS
- Šířka komplexu QRS
- Tvar komplexu QRS
- Přítomnost delta vlny
- Svod, ve kterém se nachází izoelektrická linie EKG signálu
- Svod, ve kterém se nachází nejvyšší R-vlna

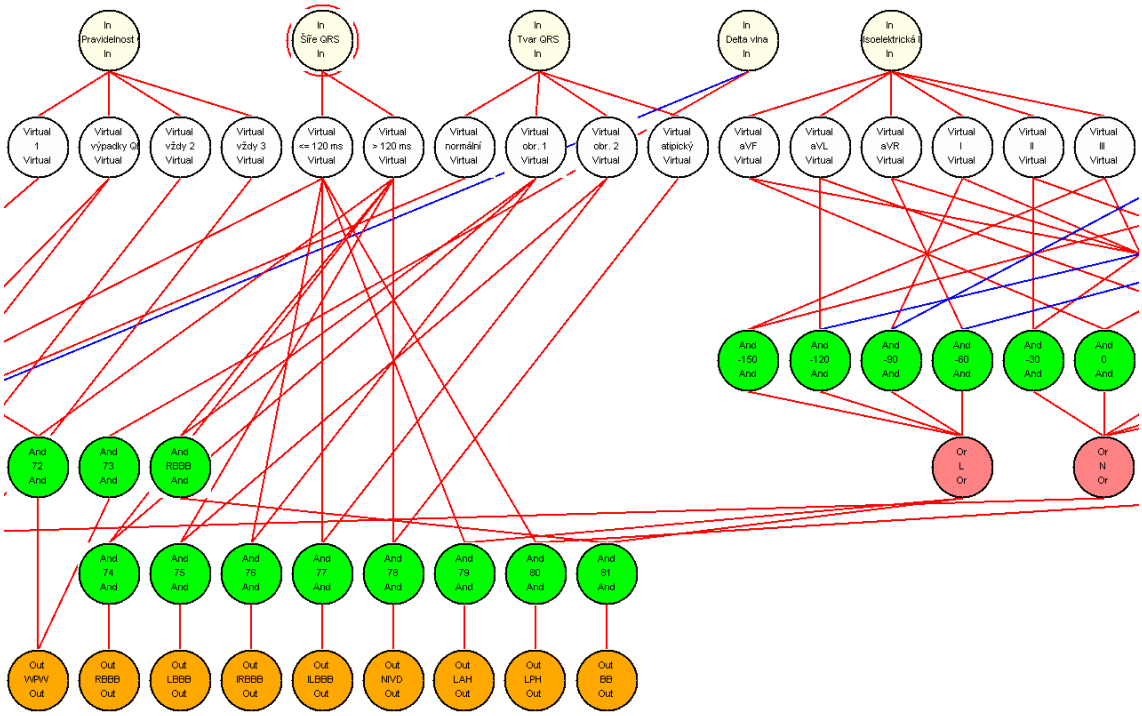
Uvedené parametry jsou vázány na příslušný svod EKG, pokud je to nutné.

Báze má 15 cílových uzlů zastupujících tyto hypotézy:

- Normální EKG
- AV blok 1. stupně
- AV blokáda 2. stupně Mobitzova typu
- AV blokáda 2. stupně Wencenbachova typu
- AV blokáda 2. stupně převod 2:1
- AV blokáda 2. stupně převod 3:1
- WPW syndrom
- Blokáda pravého Tawarova raménka (RBBB)
- Blokáda levého Tawarova raménka (LBBB)
- Inkompletní blokáda pravého Tawarova raménka (IRBBB)
- Inkompletní blokáda levého Tawarova raménka (ILBBB)
- Levý přední hemiblok
- Levý zadní hemiblok
- Bifascikulární Blokáda

Vlastní struktura báze znalostí diagnostiky bradyarytmií je tvořena dalšími 60 uzly, které tvoří 32 pravidel ve třech vrstvách. Pravidla jsou navázány na agregační funkce, které zastupují příslušné hypotézy. Jelikož se v bázi znalostí vyskytují cílové uzly, které mají být platné pouze při splnění všech předpokladů (alespoň částečném, tedy hodnota vstupů pravidla různá od 0), jsou vnitřní uzly nejprve vyhodnoceny jedním konjunktivním pravidlem a následně navázány na agregační funkci. Tento systém se pak chová jako vážená logická funkce. V tomto případě by bylo možné cílovou agregační funkci vynechat a použít jako cílový uzel přímo poslední pravidlo navázané na agregační funkci. Analyzátor expertního systému RESLA automaticky identifikuje, které uzly jsou cílové. Cílový uzel pro WPW syndrom se má chovat takovým způsobem,

že každé, byť částečné, splnění předpokladů vedoucích k tomuto cíli má postupně ovlivňovat výslednou pravděpodobnost. Proto jsou vnitřní uzly (pravidla) navázány na cílové agregační pravidlo separátně. Zde je jeho použití nutné. Ukázka grafické podoby části báze znalostí přejaté z programu RESLA je na obrázku 21. Kompletní grafická podoba báze znalostí je uvedena v příloze A.



Obrázek 21: Grafická reprezentace části báze znalostí pro diagnostiku poruch srdečního rytmu

Báze znalostí pro diagnostiku bradyarytmií byla vytvořena na základě studia literatury [11], [12], [23] a konzultacemi s lékařem. Tato báze byla vytvořena pro účely testování algoritmu ladění vah pravidel v bázi znalostí a ačkoliv je klinicky správná, nezahrnuje všechny možné poruchy EKG.

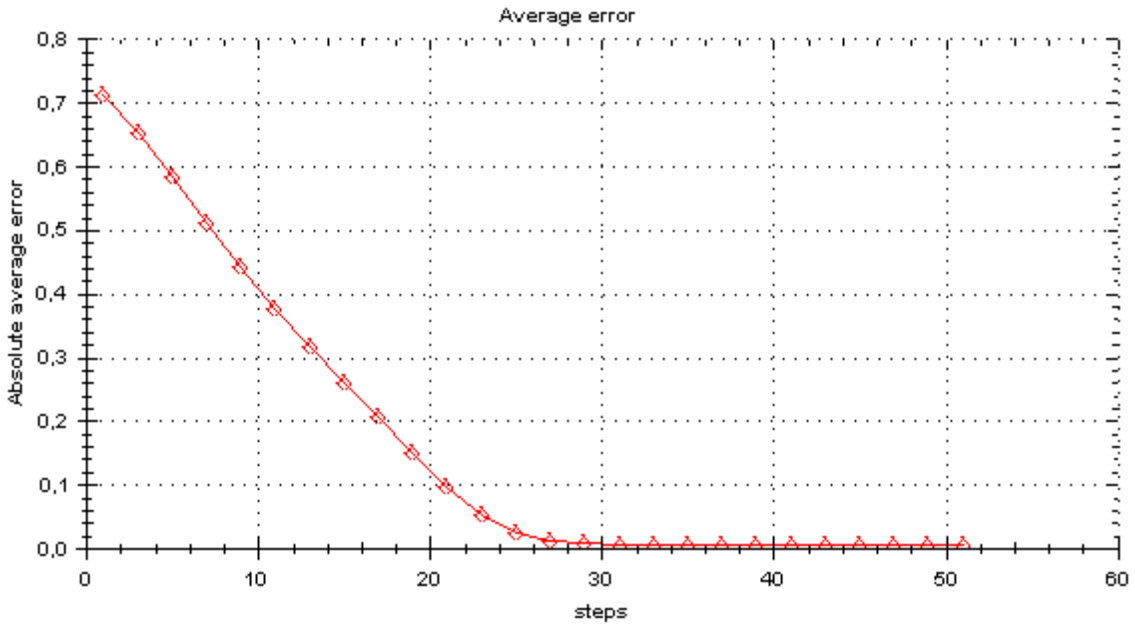
6.2.3 Tvorba vzorů a učení báze znalostí

Vzory pro trénování báze znalostí byly vytvořeny na základě vzorových signálů EKG z literatury [12]. Trénovací vzory ukazuje tabulka 7.

Délka intervalu PR [ms]	Interval PR	Počet vln P před QRS	Šíře QRS	Tvar QRS	Delta vlna přítomna ve dvou z V1 - V6:	Isoelekt-rická linie ve svodu:	Nejvyšší R-vlna	Diagnóza
120 - 200	konstantní	1	≤ 120 ms	normální	ne	aVF	I	Normální
> 200	konstantní	1	≤ 120 ms	normální	ne	aVF	I	AV blok 1. st.
120 - 200	konstantní	výpadky QRS	≤ 120 ms	normální	ne	aVF	I	AV blok 2. st - Mobitzova bl.
120 - 200	prodlužující se	výpadky QRS	≤ 120 ms	normální	ne	aVF	I	AV blok 2. st. Wencenbachuv typ
120 - 200	alternující	vždy 2	≤ 120 ms	normální	ne	aVF	I	AV blok 2. st. převod 2:1
120 - 200	alternující	vždy 3	≤ 120 ms	normální	ne	aVF	I	AV blok 2. st. převod 3:1
≤ 120	konstantní	1	≤ 120 ms	normální	ano	aVF	I	WPW syndrom
120 - 200	konstantní	1	> 120 ms	obr.1	ne	aVF	I	RBBB
120 - 200	konstantní	1	> 120 ms	obr. 2	ne	aVF	I	LBBB
120 - 200	konstantní	1	≤ 120 ms	obr. 1	ne	aVF	I	IRBBB
120 - 200	konstantní	1	≤ 120 ms	obr. 2	ne	aVF	I	ILBBB
120 - 200	konstantní	1	> 120 ms	normální	ne	aVF	I	NIVD
120 - 200	konstantní	1	≤ 120 ms	normální	ne	I	I	Levý přední hemiblok (LAH)
120 - 200	konstantní	1	≤ 120 ms	normální	ne	aVR	III	Levý zadní hemiblok (LPH)
120 - 200	konstantní	1	> 120 ms	obr. 1	ne	I	I	Bifascikulární blokáda (BB) + RBBB

Tabulka 7: Trénovací vzory báze znalostí pro diagnostiku poruch srdečního rytmu

Báze znalostí byla naučena na 15 vzorů z tabulky 7 ve 42 učících cyklech (+10 cyklů podmínky zastavení). Průběh střední absolutní chyby během učení je možné vidět v grafu na obrázku 22. Koeficient učení byl zvolen $\mu_t = 0{,}2$.



Obrázek 22: Vývoj střední absolutní chyby v průběhu trénování báze

Střední absolutní chyba byla vybrána, oproti běžnější střední kvadratické chybě, kvůli lepší vypovídací hodnotě. Tato hodnota nám říká, jaká je střední procentuální (po vynásobení stem) odchylka na výstupech báze pravidel pro všechny vzory. Síť se na vzory naučila s přesností lepší než 1 %, můžeme tedy na všech výstupech očekávat pro učící vzory průměrnou odchylku méně než 1 %.

Nutno ovšem podotknout, že učící algoritmus vnitřně pracuje se střední kvadratickou chybou.

Testovací vzory byly vytvořeny z reálných EKG záznamů za pomoci lékaře a z literatury [17], [25] . Testovací vzory jsou uvedeny v tabulce 8.

Délka intervalu PR	Interval PR	Počet vln P před QRS	Šíře QRS	Tvar QRS	Delta vlna přítomna ve dvou z V1 - V6:	Isoelektrická linie ve svodu:	Nejvyšší R-vlna	Diagnóza
120 - 200 ms	konstantní	1	<= 120 ms	normální	ne	aVL	II	Normální
120 - 200 ms	konstantní	1	<= 120 ms	normální	ne	aVF	I	Normální
> 200 ms	konstantní	1	<= 120 ms	normální	ne	aVL	II	AV blok 1. st.
> 200 ms	konstantní	1	<= 120 ms	atipický	ne	I	aVL	AV blok 1. st. + LAH
> 200 ms	konstantní	1	> 120 ms	obr. 1	ne	II	I	AV blok 1. st. + RBBB
> 200 ms	prodlužující se	výpadky QRS	<= 120 ms	normální	ne	III	I	AV blok 2. st. Wencenbachuv typ
120 - 200 ms	alternující	vždy 2	<= 120 ms	normální	ne	aVF	I	AV blok 2. st. převod 2:1
120 - 200 ms	alternující	vždy 2	<= 120 ms	normální	ne	II	aVL	AV blok 2. st. převod 2:1
<=120 ms	konstantní	0	> 120 ms	normální	ano	aVF	I	WPW syndrom
<=120 ms	konstantní	1	> 120 ms	normální	ano	I	aVF	WPW syndrom
120 - 200 ms	konstantní	1	> 120 ms	obr. 1	ne	aVF	I	RBBB
> 200 ms	konstantní	1	> 120 ms	obr. 1	po	aVL	III	RBBB + AV blok 1. st.
120 - 200 ms	konstantní	1	> 120 ms	obr. 2	ne	aVF	I	LBBB
120 - 200 ms	konstantní	1	<= 120 ms	obr. 1	ne	III	I	IRBBB
> 120 ms	konstantní	1	<= 120 ms	obr. 1	ne	II	I	IRBBB + AV blok 1. st.
120 - 200 ms	konstantní	1	<= 120 ms	obr. 2	ne	aVF	I	ILBBB
120 - 200 ms	konstantní	1	<= 120 ms	obr. 1	ne	aVR	I	Levý přední hemiblok (LAH)
120 - 200 ms	konstantní	1	<= 120 ms	normální	ne	I	aVL	Levý přední hemiblok (LAH)
> 200 ms	konstantní	1	<= 120 ms	normální	ne	I	III	Levý zadní hemiblok (LPH) + AV blok 1. st.
120 - 200 ms	konstantní	1	> 120 ms	obr. 1	ne	aVR	aVL	Bifascikulární blokáda (BB) + RBBB

Tabulka 8: Testovací vzory báze znalostí pro diagnostiku poruch srdečního rytmu

Naučená báze znalostí byla testována na 20 testovacích EKG záznamech. Průměrná chyba přes všechny testovací vzory a všechny výstupy byla 2 %. Největší odchylka činila pro jeden testovací vzor 12 %, nejmenší odchylka přes všechny vzory <1 %, medián <1 %. Z 12 % odchylky daného výstupu plyne, že jeho pravidla nejsou vhodně pokryta trénovacími vzory. Pro přesnější výsledky je nutné doplnit trénovací množinu báze znalostí.

6.3 Báze znalostí pro výběr pšenice ozimé

Báze znalostí představuje systém pro podporu při výběru vhodné odrůdy pšenice ozimé pro pěstování v různých lokalitách a různých klimatických podmínkách. Úlohou třetího testu bylo porovnat přístupy k tvorbě bází znalostí a schopnosti generalizace báze u profesionální báze znalostí vytvořené ručně expertem a báze znalostí odladěné algoritmem přímého ladění vah pravidel. Pro tuto úlohu byla vybrána báze znalostí Pšenice ozimá 2003, vytvořená experty z oblasti zemědělství Ing. Zdeňkem Kryštofem a Ing. Janem Ksiażkiewiczem, CSc. v expertním systému NPS32 a odladěna ve spolupráci DITANA spol. s r.o., Velká Bystřice. Báze znalostí obsahuje 64 odrůd povolených pro pšenici obecnou ozimou v roce 2003.

6.3.1 Struktura báze znalostí pro výběr pšenice ozimé

Báze znalostí pro podporu při výběru vhodné odrůdy pšenice ozimé měla být vytvořena transformací struktury pravidel použitých v bázi znalostí expertního systému NPS32 na bázi znalostí expertního systému RESLA, tak, aby mohly být porovnány. Ukázalo se, že v důsledku výrazně odlišného zpracování informace v pravidlech (vnitřní a výstupní uzly) báze znalostí v expertním systému NPS32 vůči expertnímu systému RESLA, není možné báze znalostí přímo transformovat. Přímá transformace báze znalostí nelze také provést, neboť původní báze znalostí obsahuje paralelní větvení při inferenci dělené kontextovými vazbami, které ovšem expertní systém RESLA neimplementuje. Z tohoto důvodu byly transformovány pouze vstupní uzly, dotazovatelné kvantitativní uzly (viz. [44]) a cílové uzly. Struktura vnitřních uzlů byla přepracována ručně na základě původních informací od experta. Vzhledem k rozsahu báze znalostí a nutnosti bázi přepracovat ručně, bylo náhodně vybráno 10 cílových uzlů (možných odrůd pšenice ozimé), které byly zpracovány v nové bázi znalostí expertního systému RESLA. Báze znalostí Pšenice ozimá 2003 byla taktéž upravena tak, aby poskytovala výsledky pouze pro vybrané cílové uzly. Informace o struktuře báze znalostí poskytnutých expertem jsou vyjádřeny tabulkou 9.

Název odrůdy	**Zemědělská výrobní oblast**																
	Kukuřičná	**Intenzita pěstování**			**Řepařská**	**Intenzita pěstování**			**Obilnářská**	**Intenzita pěstování**			**Bramborářská**	**Intenzita pěstování**			**Pícninářská**
		V	**S**	**N**		**V**	**S**	**N**		**V**	**S**	**N**		**V**	**S**	**N**	
Alka	**-**	-	-	-	**x**	xx	xx	x	**xx**	xx	xx	-	**xx**	x	x	x	**x**
Bill	**x**	xx	-	-	**xx**	xx	xx	-	**xx**	xx	xx	-	**xx**	xx	xx	-	**-**
Bruta	**x**	-	xx	xx	**x**	-	xx	xx	**x**	-	-	-	**-**	-	-	-	**-**
Ebi	**-**	-	-	-	**xx**	x	xx	xx	**xx**	xx	xx	-	**xx**	x	x	x	**x**
Estica	**x**	-	-	-	**xx**	-	xx	xx	**xx**	xx	xx	-	**x**	-	-	-	**-**
Ilias	**x**	xx	xx	-	**xx**	xx	xx	x	**xx**	xx	xx	x	**xx**	xx	xx	-	**-**
Samara	**-**	-	-	-	**x**	xx	xx	-	**xx**	xx	xx	-	**xx**	xx	x	-	**-**
Siria	**-**	-	-	-	**x**	xx	xx	-	**xx**	xx	xx	-	**xx**	xx	x	-	**-**
Trane	**-**	-	-	-	**xx**	xx	xx	-	**xx**	xx	xx	-	**xx**	xx	xx	-	**-**
Windsor	**-**	-	-	-	**xx**	xx	xx	-	**xx**	xx	xx	-	**xx**	xx	xx	-	**-**

pokračování na další straně

Název odrůdy	Regist.od roku	Pekař. jakost	Půda	Zimo-vzdor.	Předč. výsev	Před- plodina	Přísu- šek	Stéblo -lam	Padlí tr. list	Rez pšenič.	Branič. list	Poleh	Vysvětlivky
	Zač.	E +3	Lehká	Vo +3	Ne	Výbor.	Snáší	Vo +3 (8-9)					E- elitní
	+3	A +2	+3	O +2	0	+3	-3	O +2 (7)					A - kvalitní
	1995	B +1	Střed	So +1	Ano	Prům.	Toler.	So +1 (6)					B - chlebová
	0	C -3	0	Mn -1	+3	0	0	Mn -1(5)					C - krmná
	2000		Těžká	N -2		Obil.	Nesn.	N -2(4)					
	-3		-3	Vn -3		-3	+3	Vn -3(3 a méně)					Vo - vysoce odolná
Alka	0	+2	-3	-2	0	+2	+3	-3	-1	+1	+1	-1	O - odolná
Bill	-3	+2	0	-3	0	+3	+3	+1	+1	+1	+1	+1	So - středně odolná
Bruta	+3	+2	+3	+2	0	+1	-1	-1	+1	+2	+1	-1	Mn - mírně náchylná
Ebi	0	+3	+3	+2	0	-2	+2	+1	-2	-2	-1	+2	N - náchylná
Estica	0	-3	-3	-1	0	-2	+3	+3	+1	+3	+1	+1	Vn - vysoce náchylná
Ilias	-3	+2	0	+1	0	-3	0	+1	+1	+2	+1	+1	
Samara	0	-3	-3	-1	0	+2	+3	-2	-3	-2	-2	+1	
Siria	+3	+1	-3	-2	0	-2	+3	-1	-2	-1	-1	-1	
Trane	+3	-3	0	-2	0	+1	+3	+3	+2	+1	+3	+1	
Windsor	-3	-3	0	-3	+3	+2	+2	+2	+1	+1	+1	-1	

Tabulka 9: Původní popis báze znalostí Pšenice ozimá 2003

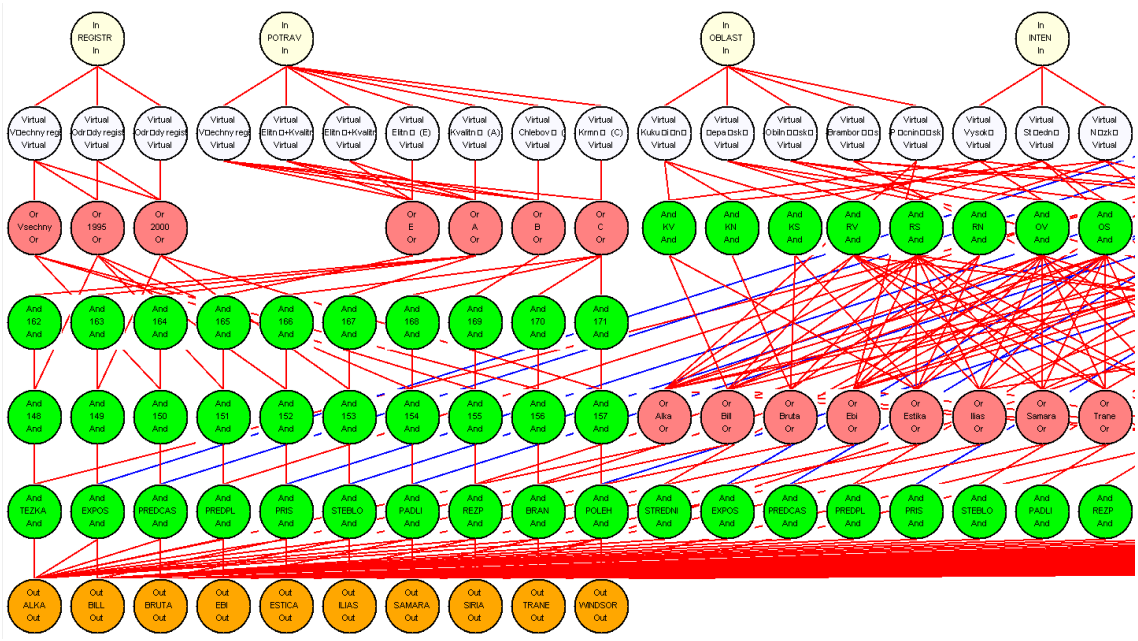
Báze znalostí se skládá ze 14 dotazovatelných vstupních uzlů zastupujících tyto parametry:

- Upřednostňovaná skupina odrůd dle roku registrace
- Požadovaná pekařská jakost
- Klimatický region
- Intenzita pěstování
- Typ půdy
- Zimovzdornost
- Ranný výsev
- Kvalita předplodiny
- Pravděpodobnost přísušku v závěru vegetace
- Pravděpodobnost stéblolamu
- Pravděpodobnost výskytu padlí travního
- Pravděpodobnost výskytu rzi pšeničné
- Pravděpodobnost výskytu Braničnatek listových
- Pravděpodobnost polehu porostu

Na základě těchto parametrů jsou vyhodnocovány nejvhodnější odrůdy pšenice ozimé. Ze 64 možných odrůd bylo náhodně vybráno těchto deset odrůd:

- Alka
- Bill
- Bruta
- Ebi
- Estika
- Ilias
- Samara
- Siria
- Trane
- Winsdor

Báze znalostí pro podporu při výběru vhodné odrůdy pšenice ozimé je tvořena dalšími 160 uzly, které tvoří 149 pravidel. Všechny výstupní uzly báze znalostí jsou realizovány agregačními funkcemi, které agregují postupně získávanou informaci z pravidel při zodpovídání otázek zastoupených dotazovatelnými uzly. Grafická podoba části báze znalostí přejatá z expertního systému RESLA je na obrázku 23.



Obrázek 23: Grafická reprezentace části báze znalostí pro výběr vhodné odrůdy pšenice ozimé

6.3.2 Tvorba vzorů a učení báze znalostí

Trénovací vzory pro bázi znalostí byly získány za pomoci konzultací v expertním systému NPS32 s bází znalostí Pšenice ozimá 2003. Konzultace byly prováděny tak, aby daný vzor, tedy hodnoty odpovědí na dotazovatelné uzly, co nejvíce podporoval vybraný cílový uzel. Takto byly stanoveny trénovací vzory pro všechny cílové uzly. Jelikož vzory nevylučují podporu i ostatních cílových uzlů (přestože byl vzor tvořen pro jiný cílový uzel) je každý cílový uzel zastoupen několika (přibližně 15) vzory. Základní množina deseti vzorů je uvedena v tabulce 9 a 10. Tabulka 9 reprezentuje odpovědi na dotazovatelné uzly, tabulka 10 zobrazuje hodnoty výstupů báze znalostí pro každou množinu těchto odpovědí. Kompletní výpis všech 23 vytvořených vzorů je možné vidět v modulu pro tvorbu trénovacích vzorů expertního systému RESLA.

vzor	Regist.od roku	Pekař. jakost	Zemědělská výrobní oblast	Intenzita pěstování	Půda	Zimo-vzdor.	Předč. výsev
1	1995	A	Obilnářská	vysoká	těžká	Spíše ne	Ne
2	2000	A	Řepařská	vysoká	střední	Ne	Ne
3	Všechny	A	Kukuřičná	nízká	lehká	Spíše ano	Ne
4	1995	E	Řepařská	(střední)	těžká	Spíše ano	Ne
5	1995	C	Obilnářská	střední	těžká	Asi ne	Ne
6	2000	A	Obilnářská	vysoká	střední	Snad ano	Ne
7	1995	C	Bramborářská	vysoká	těžká	Spíše ne	Ne
8	Všechny	B	Řepařská	střední	těžká	Spíše ne	Ne
9	Všechny	C	Bramborářská	vysoká	střední	Spíše ne	Ne
10	2000	C	Řepařská	vysoká	střední	Ne	Ano

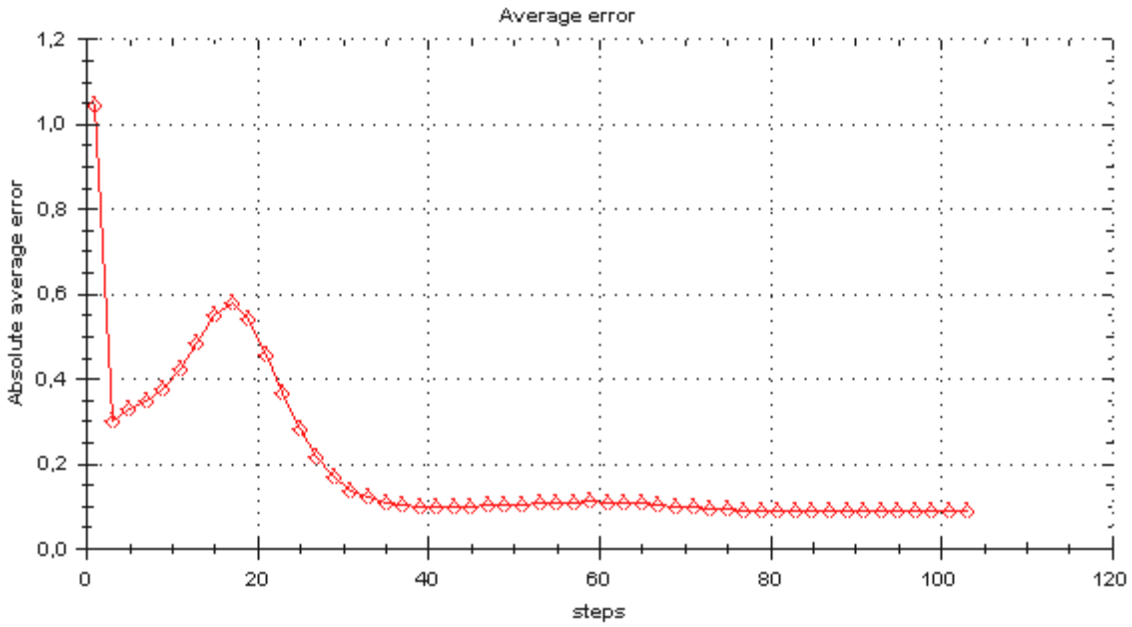
pokračování							
vzor	Před-plodina	Přísušek	Stéblolam	Padlí tr. list	Rez pšenič.	Branič. list	Poleh
1	Ano	Ne	Ne	Ne	Spíše ano	Spíše ano	Asi ne
2	Ano	Ne	Spíše ano	Spíše ano	Spíše ano	Spíše ano	Spíše ano
3	Ano	Asi ne	Asi ne	Snad ano	Spíše ano	Snad ano	Asi ne
4	Ne	Spíše ano	Snad ano	Spíše ne	Spíše ne	Asi ne	Spíše ano
5	Nevím	Ne	Ano	Snad ano	Ano	Snas ano	Snad ano
6	Ne	Snad ano	Snad ano	Snad ano	Spíše ano	Snad ano	Snad ano
7	Ne	Ne	Spíše ne	Ne	Spíše ne	Spíše ne	Snad ano
8	Ne	Ne	Asi ne	Spíše ne	Asi ne	Asi ne	Asi ne
9	Nevím	Ne	Ano	Spíše ano	Snad ano	Ano	Snad ano
10	Nevím	Spíše ano	Spíše ano	Snad ano	Snad ano	Snad ano	Asi ne

Tabulka 10: Trénovací vzory báze znalostí pro podporu pěstování pšenice ozimé – vstupy

hypotéza \ vzor	Alka	Bill	Bruta	Ebi	Estika	Ilias	Samara	Siria	Trane	Windsor
1	83,6	69,8	0,4	1,9	5,8	68,3	3,3	0	0	2,3
2	0,6	65,8	0,4	0	0	60,6	0	0	0	2,1
3	0,1	13	70,5	0	0,3	20,1	0	0	0	0
4	2,3	3	0	62,9	2,5	5,8	2,5	0	0	2,2
5	2,7	2	0	1,2	83,4	2	68,3	0	2,3	66,1
6	2,8	83,4	0,5	0	0	87,6	0	0	0	3,5
7	10,2	10,5	0	2,9	53,2	9,9	94,1	0,1	6,2	88,7
8	9	11,7	3,2	3,8	7,4	10,4	9	81,9	6,4	8
9	1	2	0	0,9	24,1	1,8	47,3	1	72,5	66,5
10	0	1	0	0	0,2	1	0,3	0	0,9	67,8

Tabulka 11: Trénovací vzory báze znalostí pro výběr odrůdy pšenice ozimé – výstupy

Báze znalostí byla naučena na 23 vzorů v 93 učících cyklech (+10 cyklů podmínky zastavení). Průběh střední absolutní chyby během učení je možné vidět v grafu na obrázku 24. Koeficient učení byl zvolen $\mu_t = 0{,}15$.



Obrázek 24: Vývoj střední absolutní chyby v průběhu trénování báze

Chyba báze znalostí po naučení trénovacích vzorů byla 0,09, tedy 9 %.

Testovací vzory byly vytvořeny, stejně jako trénovací vzory, konzultacemi v expertním systému NPS32 s bází znalostí Pšenice ozimá 2003. Základní charakteristiky odrůd pšenice ozimé (upřednostňovaná skupina odrůd dle roku registrace, požadovaná pekařská jakost a zimovzdornost), se silným vlivem na výsledné hypotézy byly voleny s ohledem na tabulku 9. Ostatní parametry byly voleny náhodně. Testovací vzory jsou uvedeny v tabulce 11 a 12. Tabulka 11 reprezentuje odpovědi na dotazovatelné uzly, tabulka 12 hodnoty výstupů báze znalostí pro každou množinu těchto odpovědí.

vzor	Regist.od roku	Pekař. jakost	Zemědělská výrobní oblast	Intenzita pěstování	Půda	Zimo-vzdor.	Předč. výsev
1	1995	A	Bramborářská	vysoká	střední	Ano	Ne
2	2000	A	Obilnářská	střední	lehká	Asi ne	Ne
3	Všechny	A	Řepařská	nízká	lehká	Spíše ano	Ne
4	1995	E	Bramborářská	vysoká	střední	Ano	Ne
5	1995	C	Řepařská	nízká	těžká	Ne	Ne
6	2000	A	Kukuřičná	vysoká	střední	Snad ano	Ne
7	1995	C	Obilnářská	střední	střední	Ne	Ne
8	Všechny	B	Bramborářská	vysoká	těžká	Spíše ne	Ne
9	Všechny	C	Bramborářská	vysoká	lehká	Ne	Ne
10	2000	C	Obilnářská	střední	střední	Spíše ne	Ano

pokračování

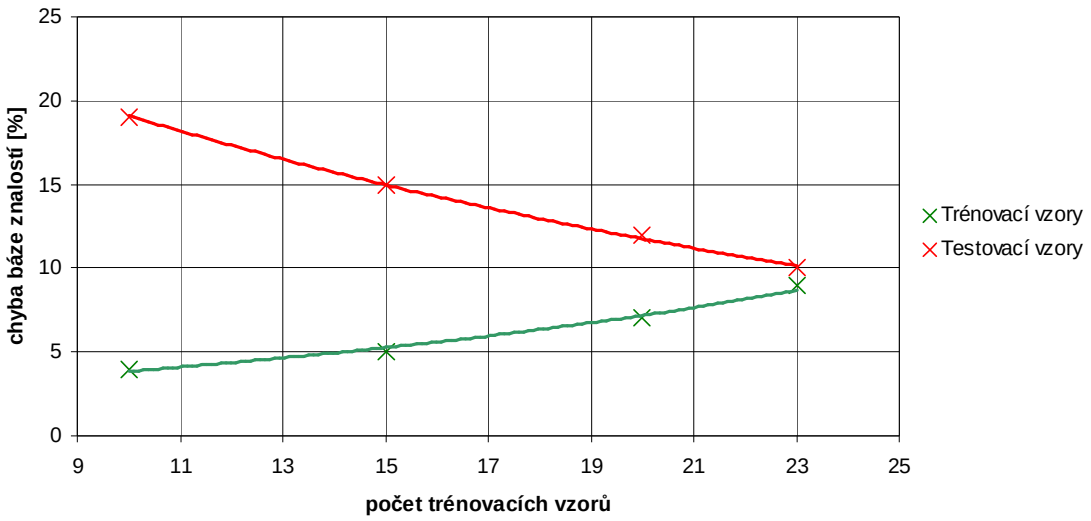
	Před-plodina	Přísušek	Stéblolam	Padlí tr. list	Rez pšenič.	Branič. list	Poleh
1	Snad ano	Asi ne	Ano	Spíše ano	Spíše ano	Spíše ano	Ne
2	Asi ne	Ne	Spíše ne	Spíše ano	Snad ano	Asi ne	Ne
3	Snad ano	Asi ne	Ano	Spíše ne	Asi ne	Snad ano	Asi ne
4	Asi ne	Nevím	Asi ne	Asi ne	Snad ano	Asi ne	Spíše ano
5	Snad ano	Ano	Snad nao	Ne	Ano	Snad ano	Ne
6	Asi ne	Snad ano	Ano	Spíše ano	Snad ano	Ano	Snad ano
7	Snad ano	Ne	Ne	Spíše ne	Asi ne	Spíše ne	Ano
8	Asi ne	Ne	Ne	Snad ano	Asi ne	Ne	Ano
9	Asi ne	Snad ano	Spíše ne	Spíše ano	Snad ano	Snad ano	Spíše ano
10	Snad ano	Spíše ne	Spíše ne	Ano	Ne	Snad ano	Spíše ne

Tabulka 12: Testovací vzory báze znalostí pro výběr odrůdy pšenice ozimé – vstupy

hypotéza \ vzor	Alka	Bill	Bruta	Ebi	Estika	Ilias	Samara	Siria	Trane	Windsor
1	30,0	34,8	0,0	0,7	0,2	60,1	0,5	0,0	0,0	0,5
2	6,2	91,2	1,6	0,0	0,0	90,9	0,1	0,0	0,1	6,3
3	34,1	26,3	66,6	4,3	1,7	61,5	0,2	0,2	0,4	0,3
4	1,5	2,5	0,0	66,2	0,3	6,8	2,5	0,0	0,0	1,8
5	1,1	0,4	0,0	1,0	75,9	1,5	21,2	0,0	0,4	31,9
6	0,0	68,0	0,3	0,0	0,0	74,4	0,0	0,0	0,0	0,0
7	2,1	3,6	0,0	3,1	79,0	2,7	70,4	0,0	3,8	79,2
8	6,1	9,6	0,0	2,3	1,1	8,5	8,4	83,3	6,2	7,0
9	2,7	7,8	0,0	2,5	28,3	7,2	72,8	1,6	83,8	86,2
10	0,0	1,4	0,0	0,0	1,3	1,2	0,8	0,0	1,5	72,9

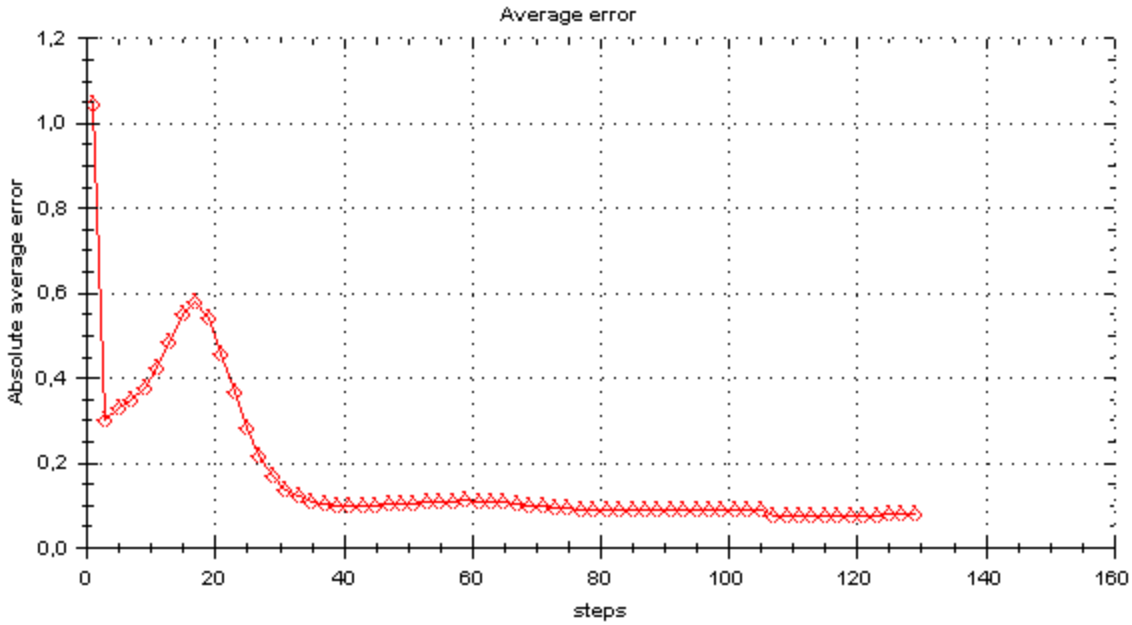
Tabulka 13: Testovací vzory báze znalostí pro výběr odrůdy pšenice ozimé – výstupy

Naučená báze znalostí byla testována na 10 testovacích vzorech. Průměrná chyba přes všechny testovací vzory a všechny výstupy byla 0,08, tedy 8 %. Pro přesnější výsledky je nutné doplnit trénovací množinu báze znalostí což dokládá graf na obrázku 25. Graf vyjadřuje závislost počtu trénovacích vzorů na průměrné absolutní chybě báze znalostí pro trénovací a testovací vzory.



Obrázek 25: Závislost průměrné absolutní chyby báze znalostí na počtu trénovacích a testovacích vzorů

Bude-li báze znalostí dále učena, dojde ke zlepšení chyby báze znalostí pro trénovací vzory, zhorší se ale chyby báze pro vzory testovací. Graf průměrné absolutní chyby po opětovném spuštění učicího algoritmu s koeficientem učení $\mu_t = 0{,}3$ je na obrázku 26.



Obrázek 26: Vývoj střední absolutní chyby v průběhu trénování báze po opětovném spuštění učicího algoritmu

Průměrná chyba přes všechny testovací vzory a všechny výstupy je nyní 0,11, tedy 11 %. Dojde k částečnému přetrénování báze. Přetrénování není ovšem tak silné jako u ANN, neboť informace nesená strukturou báze znalostí omezuje stupeň volnosti hledaných vah.

Trénovací i testovací sady vzorů vykazují vyšší chybu než tomu bylo u předchozích dvou úloh logické báze znalostí a báze pro diagnostiku arytmií. Je to způsobeno dvěma faktory. Prvním faktorem je proces získávání vzorů. Vzory byly vytvořeny konzultacemi jiného expertního systému, který uplatňuje zcela odlišné zpracování neurčitostí a práci s neurčitostí. Snadno tak nastane případ, kdy stejné struktury pravidel poskytují odlišné výstupy pro malé změny vstupních hodnot pravidel. Je to způsobeno různými vnitřními funkcemi pravidel. Druhým faktorem je podobnost struktury obou srovnávaných bází znalostí. Hlubším studiem původní báze znalostí bylo zjištěno, že struktura báze znalostí Pšenice ozimá 2003 neodpovídá zcela přesně popisu této báze znalostí, tak jak byla uvedena expertem v tabulce 9, podle které byly vytvářena struktura pro bázi znalostí expertního systému RESLA. Posledních šest faktorů vstupujících do této báze znalostí (pravděpodobnost přísušku v závěru vegetace, pravděpodobnost stéblolamu, pravděpodobnost výskytu padlí travního, pravděpodobnost výskytu rzi pšeničné, pravděpodobnost výskytu Braničnatek listových, pravděpodobnost polehu porostu) mají podle tabulky 9 kladný i záporný vliv na cílové uzly, zatímco ve skutečnosti se uplatňují všechny tyto faktory na cílové uzly záporně.

7 Programové zpracování

Kapitola popisuje programové zpracování algoritmů přímého ladění vah pravidel v bázi znalostí a jejich implementaci do expertních systémů. Kapitola je rozdělena na dvě části. První část popisuje programové zpracování metody učení s evolučním algoritmem a implementaci tohoto algoritmu do existujícího expertního systému NPS32. Druhá kapitola popisuje programové zpracování původního expertního systému RESLA vytvořeného v rámci této disertační práce.

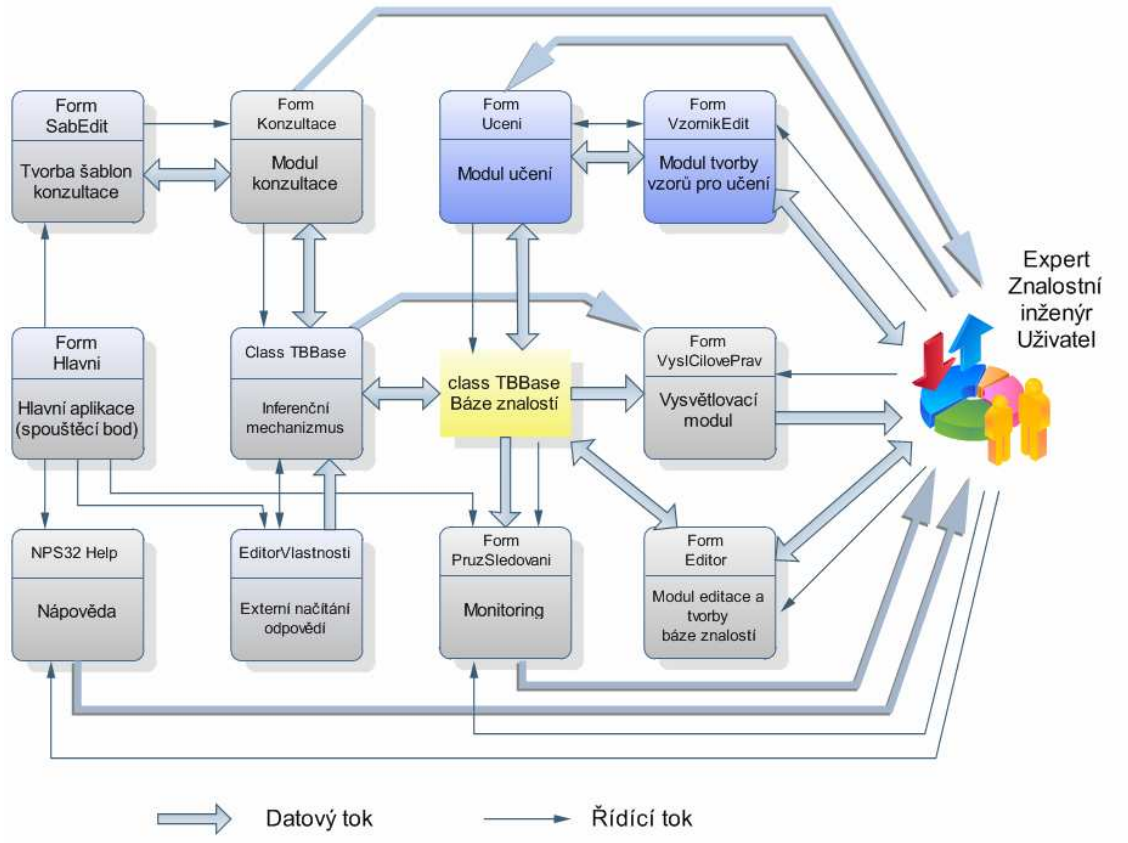
Popsáno je programové zpracování jednotlivých modulů expertních systémů, algoritmů ladění vah, struktury programu, důležité datové objekty, životní cyklus báze znalostí a ovládání programu.

7.1 Expertní systém NPS32 – Ladění vah báze znalostí s evolučním algoritmem

Diagnostický expertní systém NPS vznikl na Ústavu automatizace a měřicí techniky (dále jen UAMT), Fakultě elektrotechniky a komunikačních technologií VUT v Brně již v roce 1987. V roce 2001 byl přepracován v programovacím prostředí Borland C++ Builder [44] do současné podoby expertního systému NPS32. Expertní systém slouží doposud na UAMT ke studijním účelům, zejména k výuce tvorby bází znalostí.

Program tvoří 11 modulů, které poskytují uživateli, či expertovi, možnosti tvorby a úpravy bází znalostí, konzultace, tvorbu šablon pro konzultace[3], načítání odpovědí při konzultaci z externích zdrojů. Dále pak moduly monitorování činnosti systému při konzultaci a modul vysvětlování. Tyto jmenované části tvoří původní expertní systém NPS32. Jednotlivé moduly jsou vyznačeny na obrázku (obrázek 24) struktury expertního systému šedou barvou.

[3] Šablony slouží pro ukládání předdefinovaných odpovědí na dotazy expertního systému při konzultaci tak, aby nebyly v příštích konzultacích již pokládány. Tedy předpokládá se, že by daný uživatel odpovídal na konkrétní otázku stále stejně pro jakoukoliv konzultaci.



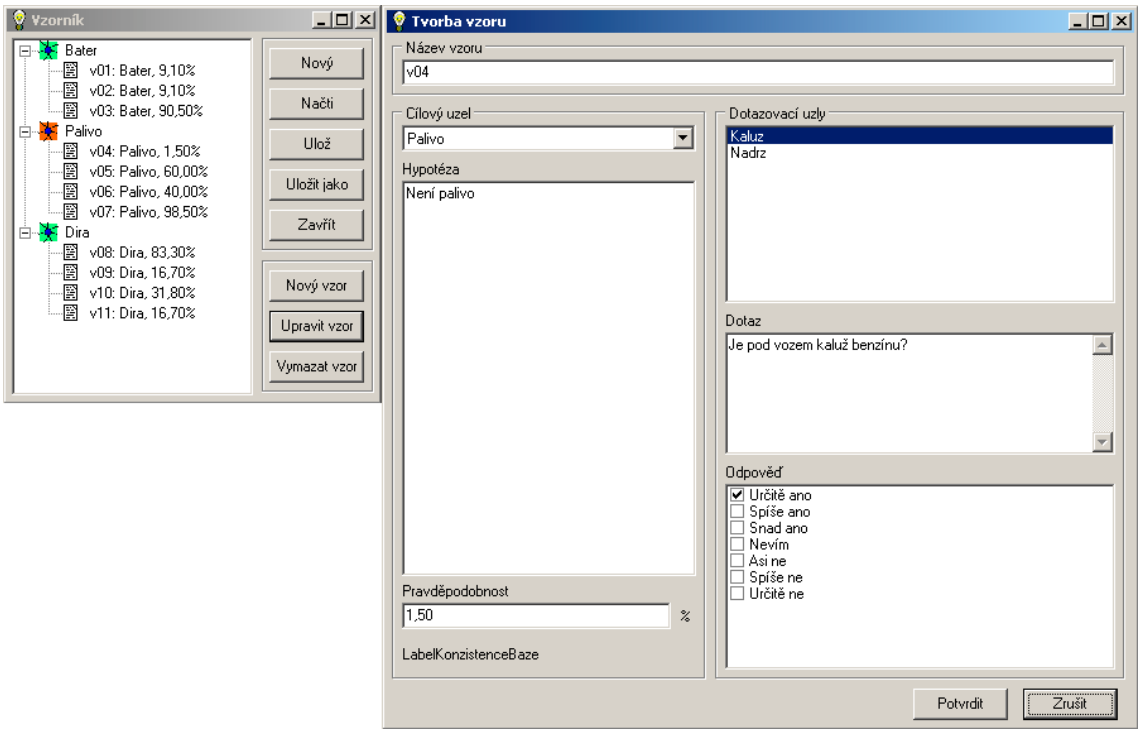
Obrázek 27: Struktura expertního systému NPS32

Modré moduly jsou nově přidané moduly učícího algoritmu s diferenciální evolucí. V těchto nových modulech učení je implementován algoritmus popsaný v kapitole 3.

Detailní informace o programovém zpracování původní části expertního systému NPS32 je možno najít ve [44].

7.1.1 Tvorba vzorů

Trénovací vzory pro diferenciální evoluční algoritmus reprezentují odezvu báze znalostí jednoho cílového uzlu na vstupní informaci příslušného podstromu pravidel. Potřebný počet vzorů je vytvořen pro každý cílový uzel (pravidlo). Každý vzor, instance třídy *TBVzor*, je tedy tvořen množinou vstupních informací, zde reprezentováno jako „*TBSeznam<TBVzorPrvek> Prvky*“ (viz. obrázek 29), a hodnotou výstupního uzlu „*float PravdepodobnostC*“. Každému vzoru je přiřazeno jméno podle názvu cílového uzlu, který zastupuje, a pořadí (viz. obrázek 28). Jednotlivé vzory jsou ukládány v seznamu „*TBSeznam<TBVzor*> Prvky*“ instance třídy *TBVzornik*, která slouží jako kolekce vzorů a kontrolních informací o bázi znalostí, ke které vzory patří, nelze je tedy zaměnit pro různé báze znalostí. Vzory jsou tvořeny v editoru modulu *Vzorník* a *Vzory* které jsou zobrazeny na obrázku 27.



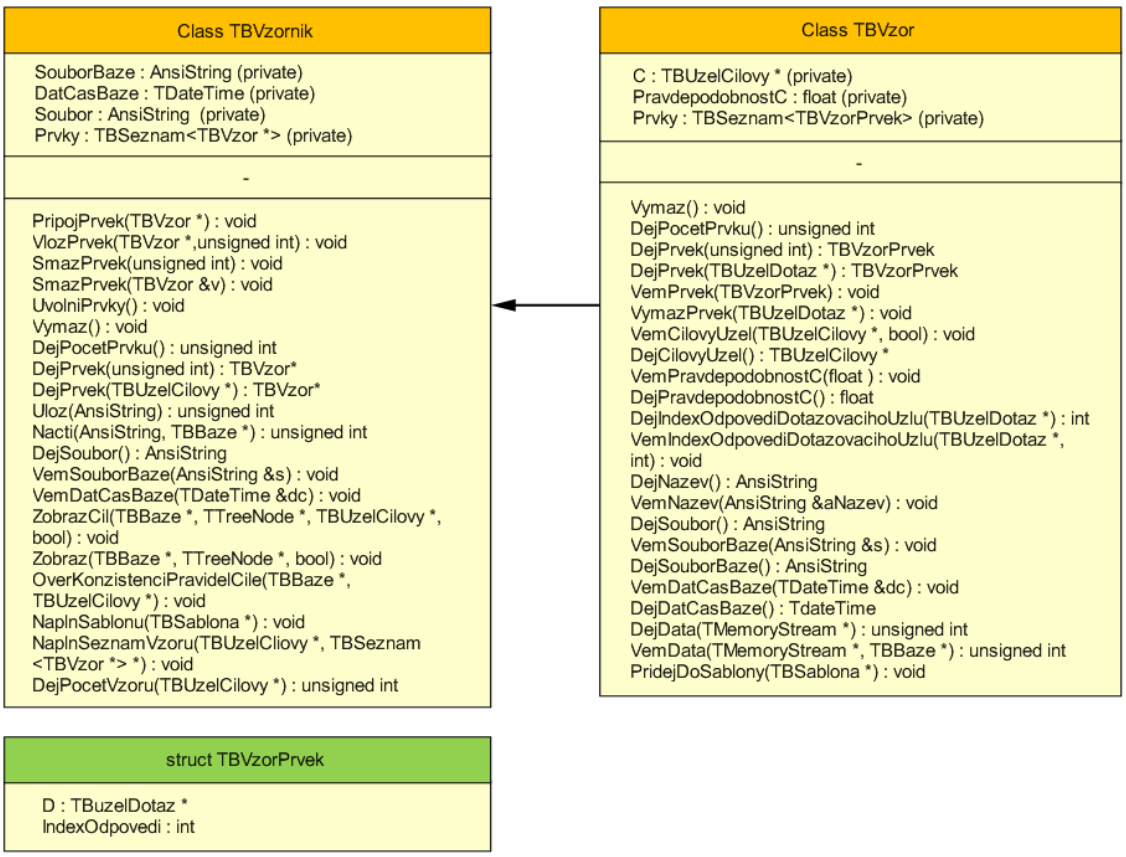
Obrázek 28: Modul tvorby vzorů

7.1.1.1 Datové struktury

Každý vzor odpovídající datům pro jeden cílový uzel je reprezentován instancí třídy *TBVzor*. Obsahuje informace o cílovém uzlu, který zastupuje (*C*), jeho požadovanou výstupní hodnotu (*PravdepodobnostC*) a seznam odpovědí na příslušné dotazovatelné uzly (*Prvky*). Tato data jsou formou privátních proměnných, tedy dostupné pouze vnitřními funkcemi třídy. Třída *TBVzor* obsahuje metody, které tvoří rozhraní ke jmenovaným privátním proměnným.

Vzory jsou vázány v kolekci vzorů *TBVzorník*, kde každému cílovému uzlu náleží určitý počet vzorů. Opět veškeré datové struktury jako cesta k souboru báze znalostí (*SouborBaze*), čas vytvoření báze (*DatCasBaze*), název souboru se vzory (*Soubor*) a jednotlivé vzory (*Prvky*) jsou privátní proměnné. Funkce definované ve třídě *TBVzornik* opět tvoří rozhraní pro manipulaci s těmito daty. Datové struktury modulu učení v UML[4] diagramu jsou zobrazeny na obrázku 29.

[4] Informace, jak číst diagramy UML, jsou uvedeny v příloze C



Obrázek 29: Datové struktury vzorů

7.1.2 Modul učení

Učení báze znalostí probíhá podle algoritmu popsaného v kapitole 4.

7.1.2.1 Metody a funkce

Proces ladění vah pravidel v bázi znalostí začíná přípravou dat pro optimalizátor. Tlačítkem *Zpracuj Vzory* se spustí funkce

```
Void __fastcall TFUceni::ButtonVzoryClick(TObject *Sender);,
```

která pro každý cílový uzel vytvoří strukturu *Params* s parametry pro optimalizaci. Do této struktury se nejdříve uloží počet vzorů pro daný cílový uzel a jeho počáteční pravděpodobnost vyjádřená dvojicí hodnot (T, F). Dále do struktury uložíme i počáteční hodnotu pravidla vedoucího do cílového uzlu. Jak bylo ukázáno v kapitole 4.2, expertní systém pracuje s vlastním matematickým aparátem, kde, veškeré pravděpodobnosti jsou vyjádřeny dvojicí hodnot (True, False). Taktéž bylo ukázáno, že tento systém nevyhodnocuje pravidla prostým skládáním pravděpodobností, ale využívá změny pravděpodobnosti, z původní hodnoty na hodnotu po zodpovězení některého dotazu, pro ovlivnění výsledné pravděpodobnosti pravidla. Tyto hodnoty, tedy počáteční hodnota pravděpodobnosti pravidla a počáteční hodnoty cílového uzlu, a hodnoty pravděpodobností pravidla a cílového uzlu po položení všech dotazů jsou potřeba připravit pro optimalizaci. Pro příslušný cílový uzel se načtou vzory ze vzorníku a pro tento každý vzor se inicializuje báze znalostí, postupně se v souladu se vzorem zodpoví příslušné otázky a přepočtením báze znalostí získáme hodnotu pravidla po zodpovězení

všech příslušných dotazů. Hodnoty se uloží do struktury *Params*. Tímto způsobem se vytvoří parametry pro DE pro všechny cílové uzly.

Tlačítkem *Inicializace* zavoláme funkci

```
void __fastcall TFUceni::ButtonOptInitClick(TObject
*Sender);,
```

která pro každý cílový uzel vytvoří instanci třídy *CDE_NPS32*, neboli vlastní optimalizátor a nastaví parametry výpočtu diferenciální evoluční metody. Těmito parametry jsou:

- pravděpodobnost křížení,
- pravděpodobnost mutace,
- váhový faktor,
- mutační faktor,
- faktor křížení,
- elitismus,
- podmínka zastavení.

Vznikne tedy pole optimalizátorů s počtem rovným počtu cílových uzlů. Následně se vytvoří první populace jedinců optimalizátoru pro každý cílový uzel.

Zaškrtnutím políčka *spustit učení* zavoláme funkci

```
void __fastcall TFUceni::CheckBoxOptRunClick(TObject
*Sender);,
```

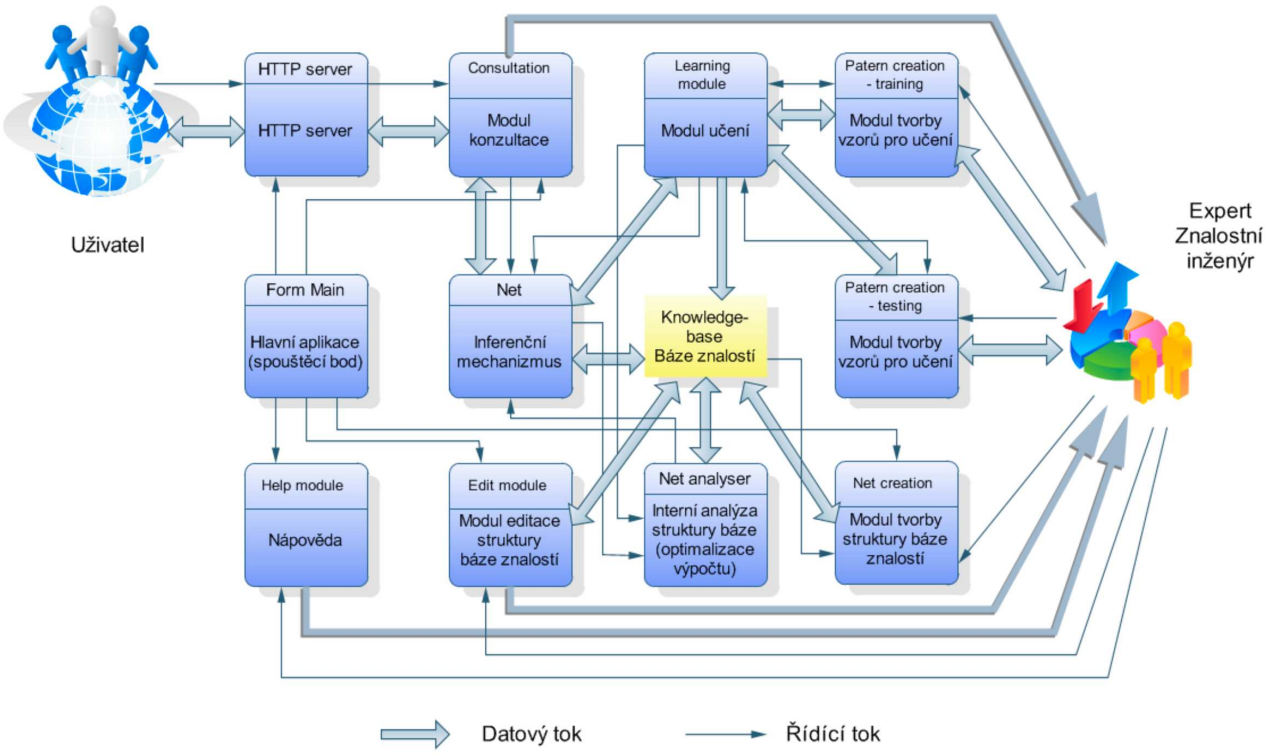
která volá postupně pro každý cílový uzel optimalizátor, který vytvoří novou populaci jedinců. Pokud nejlepší jedinec z populace zlepší řešení více než je dáno podmínkou k zastavení, daný optimalizátor bude pokračovat v činnosti, pokud podmínka k zastavení splněna není, výpočet váhy pravidla bude pokračovat. Celý výpočet se zastaví dosažením výsledných vah nebo dosažením maximálního počtu kroků u všech cílových uzlů.

7.2 Expertní systém RESLA - Metoda přímého ladění vah pravidel báze znalostí

Pro účely ověření teorie z kapitoly 5 byl vytvořen nový expertní systém RESLA (Rule-based Expert System Learning Algorithm) do nějž byl implementován algoritmus přímého učení vah báze znalostí. Varianta implementace algoritmu učení do některého ze stávajících expertních systémů vyvinutých na UAMT nebyla vhodná, neboť jde o programy vyvinuté v již zastaralém prostředí Borland Builder C++ a bylo by potřeba náročné přepracování programu. Vytvoření systému nového, který může být následně využit pro výukové účely, bylo v tomto případě efektivnější.

Expertní systém RESLA je naprogramován v programovacím jazyce C# prostředí Microsoft Visual Studio 2008 .Net. Skládá se z 10 modulů jejichž základ odpovídá standardní architektuře expertních systémů. Strukturu programu včetně řídících a datových vazeb můžeme vidět na obrázku 30.

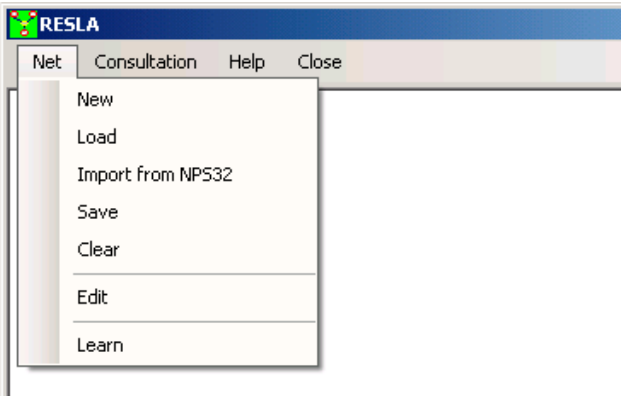
Moduly programu tvoří tři logicky a programově ucelené části a to moduly komunikace (HTTP server, Modul konzultace a Nápověda), jádro expertního systému (Báze znalostí, Interní analýza struktury báze, Inferenční mechanizmus, Modul učení) a grafické prostředí (Hlavní aplikace, Modul tvorby struktury báze, Modul editace struktury báze a oba moduly tvorby vzorů). Jednotlivé části programu jsou naprogramovány zcela odděleně s jasně definovanými rozhraními a mohou být zahrnuty do hlavního programu, nebo být součástí programu jako knihovna DLL (takto je v programu řešeno jádro expertního systému).



Obrázek 30: Struktura expertního systému RESLA

7.2.1 Hlavní aplikace

Hlavní aplikace slouží jako expertní rozhraní k programu (Uživatelské rozhraní je formou webového serveru modulu HTTP server). Obsahuje položky menu, kterými se spouští jednotlivé části programu, a barevné textové pole, které zaznamenává veškerou činnost programu a tvorbu a úpravy báze znalostí. Menu „Síť“ (Net) je zobrazena na obrázku 31.



Obrázek 31: Hlavní menu expertního systému RESLA

Hlavní menu obsahuje položky pro vytvoření nové prázdné báze znalostí (New), načtení báze z disku (Load), import z bází znalostí programu NPS32 (Import from NPS32), uložení báze na disk (Save), vymazání báze, tedy uvedení do výchozího stavu (Clear), spuštění modulu editace (Edit) a konečně spuštění modulu učení (Learn).

Menu konzultace (Consultation) řídí činnost webového rozhraní programu. Obsahuje položky spuštění webového serveru jako samostatného programu (WebServer Start) a ukončení činnosti webserveru (Stop). Server by za normálních podmínek běžel i po ukončení hlavní aplikace, proto je s ukončením hlavní aplikace automaticky také ukončen.

Menu nápovědy (Help) obsahuje informace o programu (položka About) a spuštění nápovědy (položka Help). Nápověda je realizována strukturou webových stránek.

Poslední položka menu (Close) ukončí program a webserver, pokud je spuštěn.

7.2.1.1 Metody a funkce

Hlavní aplikace obsahuje veškeré metody pro uchovávání báze znalostí. Jelikož je tato činnost závislá na platformě, je realizována zde a nikoliv v jádru expertního systému, které je na platformě nezávislé (při použití příslušného frameworku, např. .Net Framework, Mono, atd.). Je také místem uchovávání instance třídy ClassNet, tedy objektu báze znalostí a instance třídy GrNetTree, která reprezentuje její grafickou podobu.

Báze znalostí se ukládá serializací struktury, složené z instance ClassNet a instance GrNetTree, do binárního souboru. Tímto způsobem je zajištěna neporušenost báze znalostí, ať už úmyslným nebo neúmyslným zásahem, neboť soubor obsahuje různé klíče a hash kódy k rozpoznání změn a případně soubor vyhodnotí jako neplatnou bázi.

```
public void SaveNet();
```

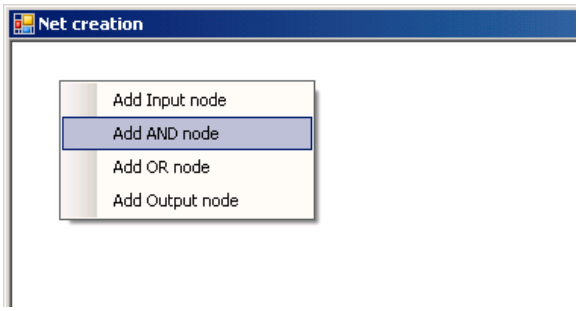
Načtení báze je inverzní operací k ukládání, tedy binární soubor je deserializován a následně rozdělen na instanci ClassNet (báze znalostí) a instanci GrNetTree (grafická reprezentace báze znalostí). Zde je realizována jako privátní funkce kterou vyvolá událost stisknutí příslušné položky menu.

```
private void loadToolStripMenuItem_Click(object sender,
EventArgs e);
```

7.2.2 Tvorba báze znalostí

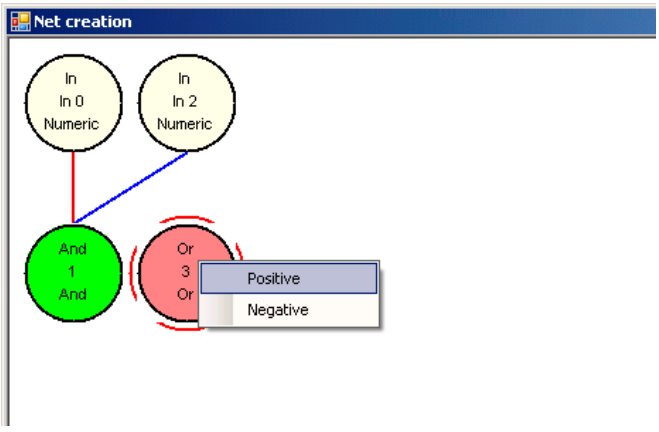
Počáteční prázdná báze znalostí je vytvořena při volbě položky „New“ z menu „Net“. Znamená to, že jsou vytvořeny instance příslušných tříd (Class_Net a GrNetTree) a v nich prázdné seznamy uzlů (pravidel), spojů a grafických prvků.

Funkční část báze znalostí je tvořena v grafickém rozhraní pomocí kontextového menu plochy modulu pro tvorbu báze znalostí (dále jen modulu). Stisknutím pravého tlačítka myši na ploše modulu je vyvoláno kontextové menu pomocí něhož můžeme vložit různá pravidla. Situace je ukázána na obrázku 32.



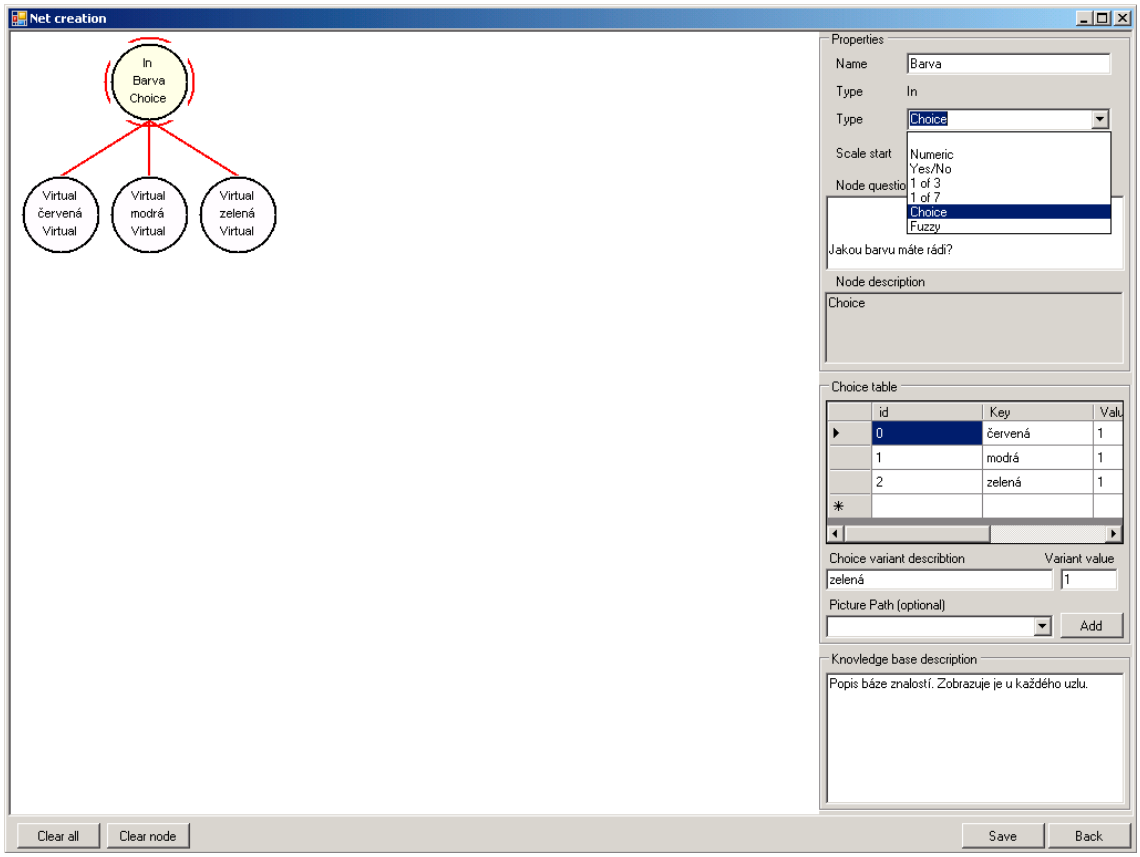
Obrázek 32: Kontextové menu pro tvorbu báze znalostí

Vložená pravidla můžeme propojit do požadované sítě (struktury báze znalostí) kliknutím pravého tlačítka myši na pravidle odkud má spoj vycházet a opětovném kliknutí pravého tlačítka myši na pravidle kde má spoj končit. Zde je vyvoláno kontextové menu (obrázek 33), ve kterém je možné zvolit typ vazby mezi pravidly (kladná, nebo záporná). Poté je spoj vykreslen.



Obrázek 33: Kontextové menu spojování pravidel v bázi znalostí

Kliknutím levého tlačítka myši na jakékoliv pravidlo se zobrazí na pravé straně pracovní plochy panel s vlastnostmi pravidla. Pokud není žádný uzel označen, panel zmizí. Pomocí tohoto panelu můžeme vlastnosti pravidla měnit. Obsah a nástroje panelu se mění v závislosti na pravidle na které je kliknuto (v programu proměnná Actual_Node). Ukázka panelu vlastností pro vstupní uzel je ukázána na obrázku 34.



Obrázek 34: Panel vlastností pravidla – vstupní uzel

Přidáním pravidla na pracovní plochu se vytvoří nová instance třídy Class_Node (viz. kapitola 7.2.2.1), a to Class_Node_In pro vstupní uzel, Class_Node_And pro konjunktivní pravidlo, Class_Node_Or pro disjunktní pravidlo a Class_Node_Out pro výstupní agregační funkci. K danému matematickému uzlu je vytvořena příslušná instance jeho grafické reprezentace (GrNode). Tyto instance jsou zařazeny do seznamu pravidel a seznamu grafických objektů v bázi znalostí. Vazba mezi těmito prvky je dána společným ID. Speciálním případem je vytváření vstupu 1 z N. Po přidání vstupního uzlu je možné na panelu vlastností zvolit položku typ uzlu (Type).

Položka ovlivňuje možnosti odpovědi uživatele na otázku zastoupenou daným vstupním uzlem. Na výběr jsou možnosti:

- Numerický vstup – zadává se jako číslo v rozsahu 0-1, případně rozsahu daném položkami měřítka (Scale) mapovanými na rozsah 0-1,
- Ano/Ne,
- 1 ze 3 – Nabízí přednastavené hodnoty dle tabulky 14,
- 1 ze 7 - Nabízí přednastavené hodnoty dle tabulky 14,
- Choice – výběr z několika možností specifikovaných při tvorbě báze znalostí,
- Fuzzy – základní fuzzyfikace, rozdělení do množin se sedmi trojúhelníkovými množinami příslušnosti (pracuje obdobně jako Choice).

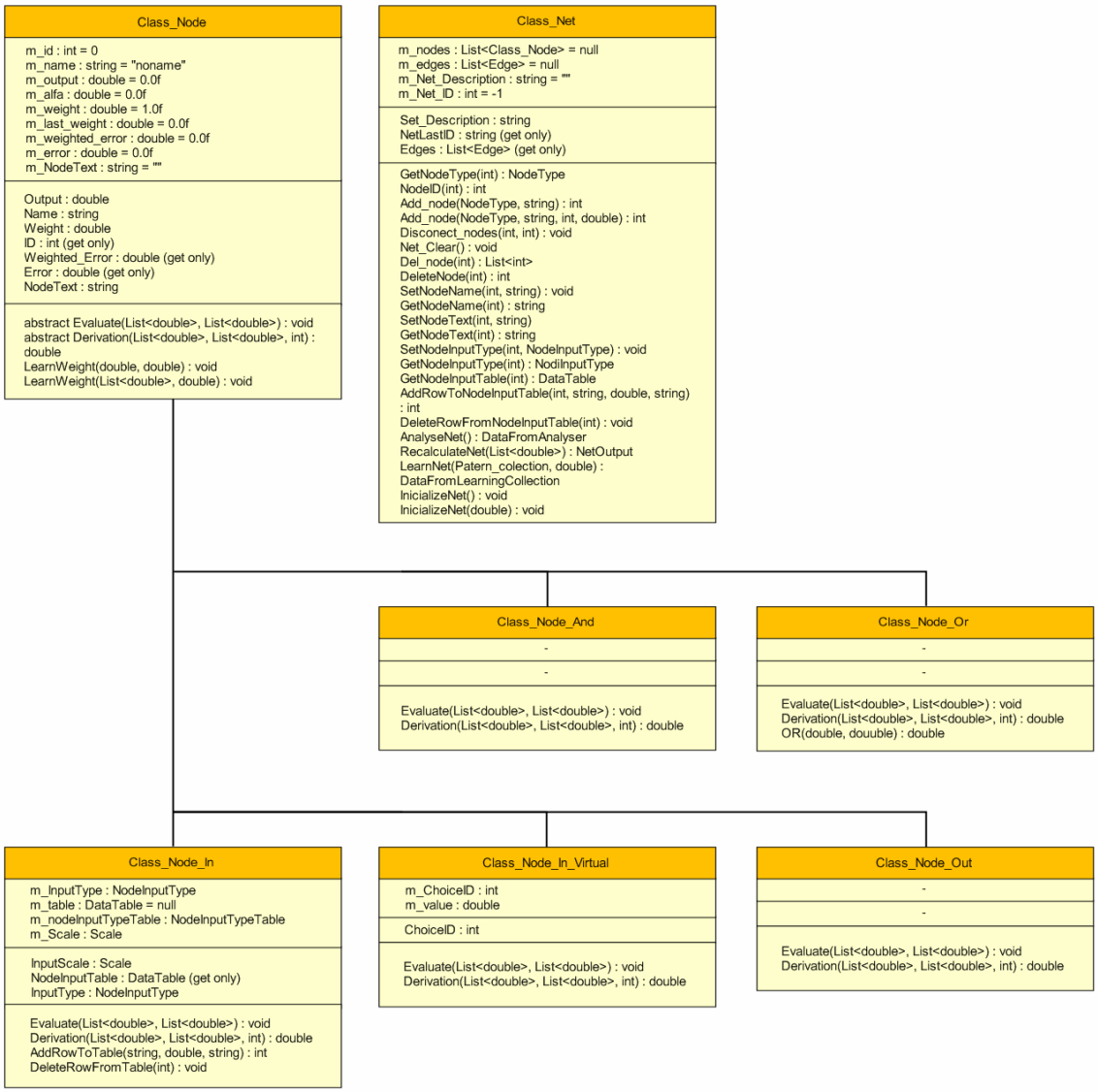
Volbou „Choice" povolíme zadávání uživatelských odpovědí. Vyplněním položky „Choice variant description" zvolíme název položky (možné odpovědi), která bude zobrazena uživateli na výběr a vyplněním položky „Variant value" zvolíme váhu daného výstupu (tato hodnota bude vstupem do struktury pravidel). Tlačítkem přidat (Add) přiřadíme tuto variantu odpovědi vstupnímu uzlu. Do báze znalostí se tak přidá instance třídy Class_Node_Virtual (viz. kapitola 7.2.2.1), která zastupuje danou odpověď. Přidá se i její grafická reprezentace (GrNode_Virtual) (viz. kapitola 7.2.2.3). Tento uzel se automaticky propojí se vstupním uzlem pozitivní vazbou. Funkce vstupního uzlu „1 z N" je taková, že vstupní uzel rozešle na všechny své následovníky (virtuální uzly) index zvolené odpovědi. V případě, že přednastavený index virtuálního uzlu je shodný s indexem odpovědi, nastaví uzel na výstup svou váhu, neboli hodnotu 1 (pravdivé tvrzení) násobenou váhou.

Text odpovědi v ES RESLA (v originále)	Text odpovědi v ES NPS32	Počet odpovědí / hodnota odpovědi					
		3			7		
		NPS32		RESLA	NPS32		RESLA
———	Ano	100%	1	———	———	———	———
Yes	Určitě ano	———	———	1	100%	3	1,0
Probably	Spíše ano	———	———	———	75%	2	0,8
Maybe	Snad ano	———	———	———	60%	1	0,6
I dont know	Nevím	50%	0	0,5	50%	0	0,5
Maybe no	Asi ne	———	———	———	40%	-1	0,4
Probably no	Spíše ne	———	———	———	25%	-2	0,2
No	Určitě ne	———	———	0	0%	-3	0,0
———	Ne	0%	-1	———	———	———	———

Tabulka 14: Srovnání hodnot pravděpodobností nabízených odpovědí a jejich slovní reprezentace expertních systémů NPS32 a RESLA

7.2.2.1 Datové struktury

Diagram na obrázku 35 zobrazuje datové třídy báze znalostí expertního systému RESLA.



Obrázek 35: Datové třídy pro tvorbu báze znalostí

7.2.2.2 Metody a funkce

Pro tvorbu báze znalostí jsou využívány funkce pro ovládání grafických prvků v programu včetně ovládání vlastních grafických objektů a funkce pro přidávání a odebírání pravidel a vazeb. Grafické funkce jsou dostatečně popsány v programu a zde uvedeny nebudou z důvodu rozsahu.

Funkce pro práci s pravidly jsou definovány ve třídě Class_Net (kapitola 7.2.2.1).

Funkce pro přidání pravidla

```
public int Add_node(NodeType aType, string aNodeName);
```

vloží instanci příslušného typu pravidla (NodeType) do seznamu pravidel sítě v instanci třídy ClassNet a přiřadí pravidlu jméno. Funkce vrací ID příslušného uzlu které je generováno automaticky jako vnitřní parametr báze znalostí.

Funkce pro přidání virtuálního uzlu

```
public int Add_node(NodeType aType, string aNodeName,
int aChoiceId, double aValue);
```

vloží instanci třídy Class_Node_In_Virtual do seznamu pravidel sítě v instanci třídy ClassNet a přiřadí uzlu jméno, ID odpovědi a váhu. Vrací ID přidaného uzlu.

Funkce pro odstranění pravidla z báze znalostí

```
public List<int> Del_node(int aID);
```

odstraní z báze znalostí pravidlo s příslušným ID, případně více pravidel (uzlů), pokud jde o vstupní uzel typu „Choice“, odstraní příslušné vazby a vrátí seznam ID smazaných uzlů.

Funkce pro vazbu pravidel

```
public void Connect_nodes(int aID1, int aID2, ConnType
aConnTtpe);
public void Disconnect_nodes(int aID1, int aID2);
```

spojí pravidla od pravidla s ID1 k pravidlu s ID2 a pozitivní nebo negativní vazbou danou typem spoje ConnType.

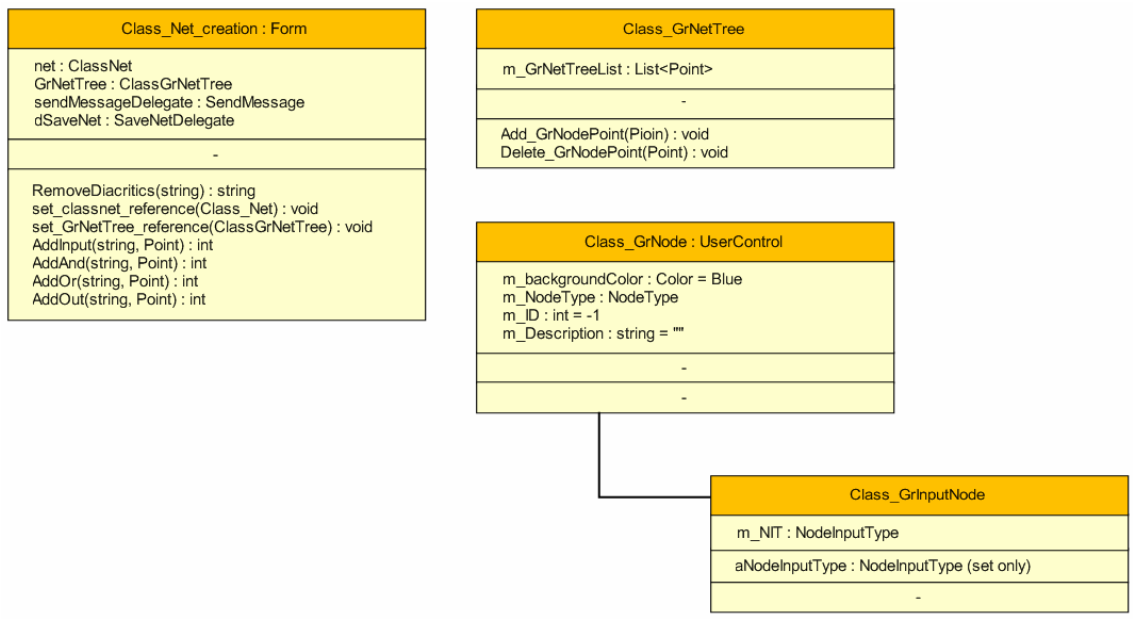
Síť je možné vymazat funkcí

```
public void Net_Clear();.
```

Třída Class_Net obsahuje dále množství funkcí pro nastavení typu pravidla a jeho vlastností, které jsou popsány v přiloženém programu.

7.2.2.3 Grafická reprezentace

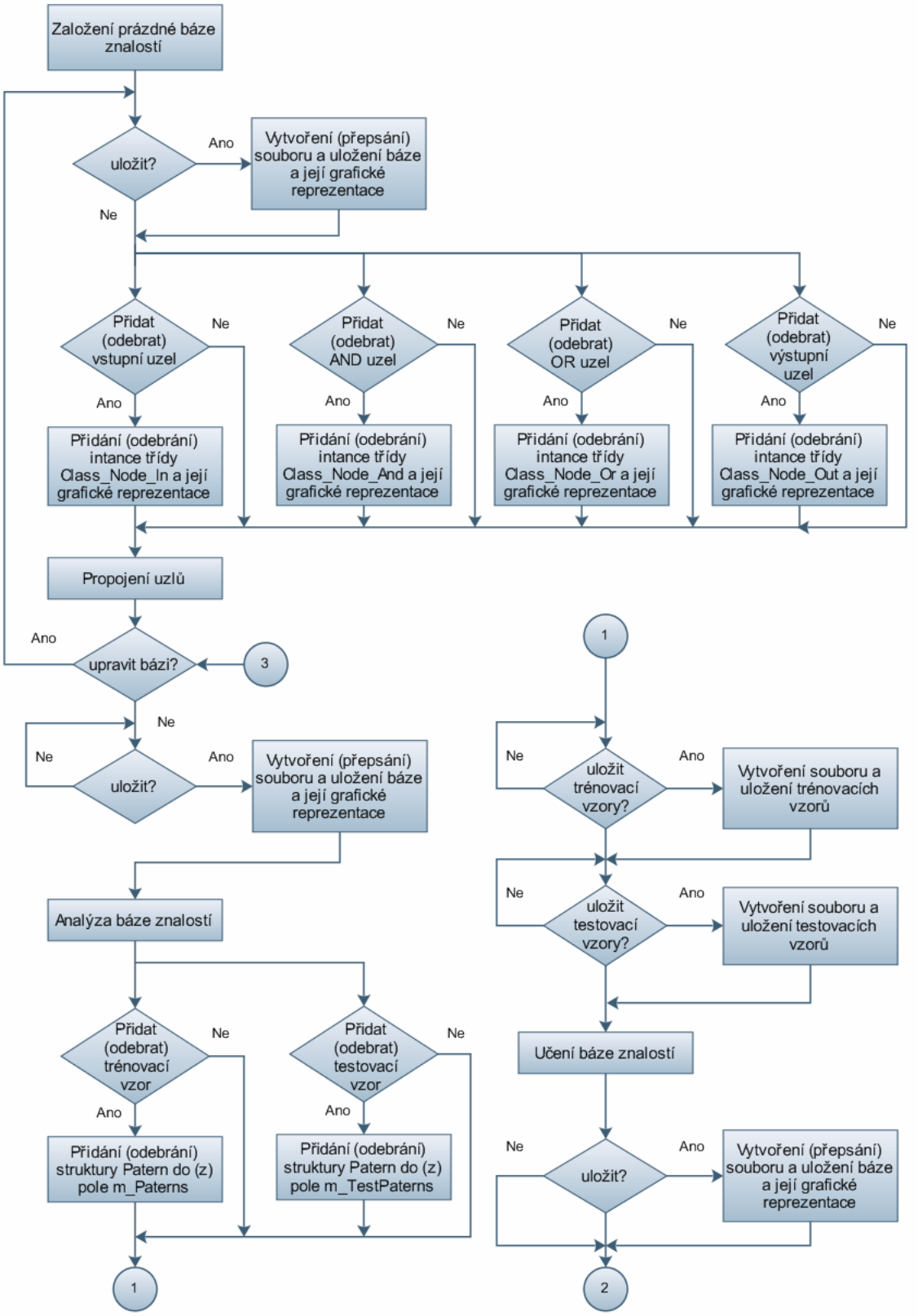
Diagram na obrázku 36 zobrazuje datové třídy grafické části báze znalostí expertního systému RESLA

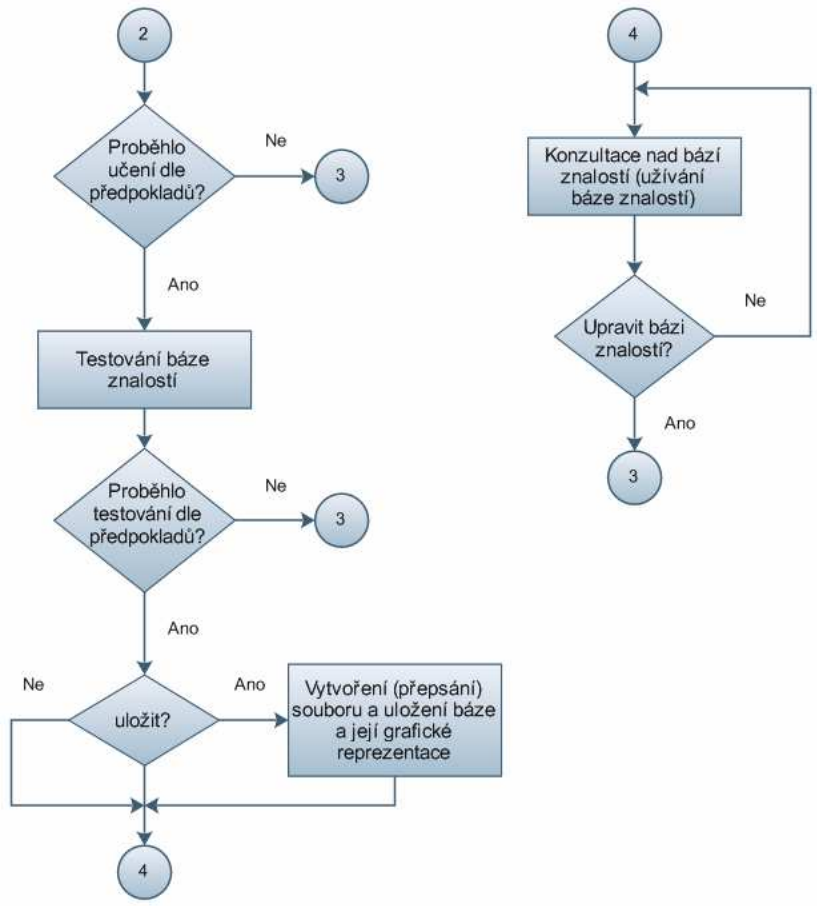


Obrázek 36: Datové třídy grafické reprezentace báze znalostí

7.2.2.4 Životní cyklus báze znalostí v expertním systému RESLA

Životní cyklus báze znalostí je dán následujícím vývojovým diagramem. Představuje postup od vzniku báze znalostí až po její používání a údržbu.





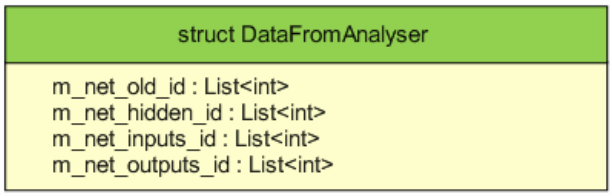
Obrázek 37: Životní cyklus báze znalostí expertního systému RESLA

7.2.3 Analyzátor báze znalostí

Analýza báze znalostí je neveřejnou, tedy privátní funkcí, třídy Class_Net. Slouží k interní analýze struktury báze znalostí, ověření její konzistence a optimalizaci výpočtu. Díky této funkci se nemusejí v bázi znalostí ukládat vazby mezi uzly, stačí pouze jejich seznam (neplatí pro učící mechanismus, tam jsou potřebné i vazby, nicméně inference je také urychlena). Báze znalostí má také větší volnost v tvorbě, neboť není specifikováno, který uzel má být na které pozici (např. uzel AND nebo OR mohou být výstupními uzly).

7.2.3.1 Datové struktury

Data z analyzátoru báze znalostí jsou ukládány ve struktuře DataFromAnalyser (obrázek 38), která jsou využívána při dopředeném přepočítávání sítě a ošetření nekonzistentnosti báze znalostí.



Obrázek 38: Struktura dat analyzátoru báze znalostí

7.2.3.2 Metody a funkce

Analýza báze znalostí

```
public DataFromAnalyser AnalyseNet();
```

vytváří strukturu čtyř polí indexů pravidel báze znalostí (instancí třídy Class_Node) na základě vzájemného vztahu těchto pravidel. Určuje, které uzly jsou vstupní, výstupní a vnitřní a chybně zapojené. Tyto indexy setřídí podle pořadí, ve kterém je nutné je přepočítat, aby báze fungovala správně. Využívá k tomu pravidel, která lze zjednodušeně charakterizovat takto:

- Vstupuje-li do uzlu vazba a zároveň z uzlu nic nevystupuje, jde o výstupní uzel,
- Vystupuje-li z uzlu vazba a zároveň do uzlu nic nevstupuje a jde o uzel Class_Node_In, jde o vstupní uzel,
- Vstupuje-li do uzlu vazba a vystupuje-li z uzlu vazba a jde o uzly Class_Node_And nebo Class_Node_Or, jde o vnitřní uzel,
- Vše ostatní je nekonzistentnost sítě (např. vystupující vazba z výstupního uzlu, vstupující vazba do vstupního uzlu, nezapojený uzel, atd.).

Analyzátor chybné uzly automaticky vyřadí z výpočtu, pokud to nenarušuje funkci báze znalostí, jinak vyzve tvůrce báze znalostí z nápravě.

7.2.4 Inferenční mechanismus

Inference nad bází znalostí je prováděna v úzké spolupráci s daty analyzátoru sítě. Funkce inference pracuje vnitřně ve třech cyklech. V těchto cyklech je pro každý vstupní uzel, každé vnitřní pravidlo a každé výstupní pravidlo (dáno strukturou `DataFromAnalyser`) postupně volána vnitřní funkce objektu pravidla, odpovídajícího indexu pole analyzátoru, k přepočítání modelu daného pravidla.

7.2.4.1 Metody a funkce

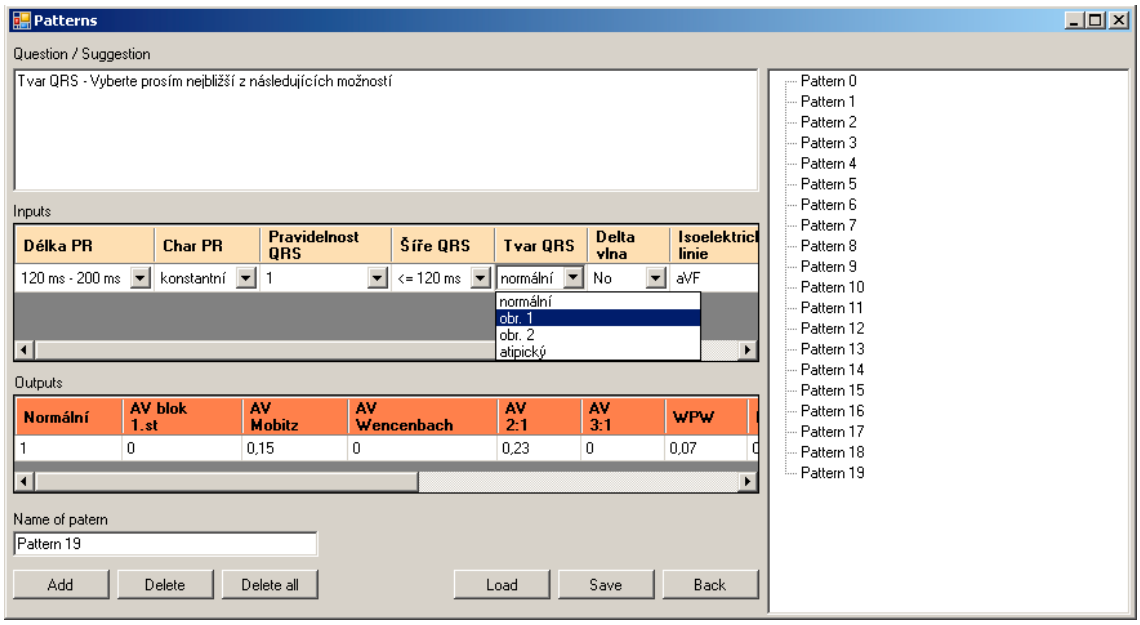
Funkce

```
public NetOutput RecalculateNet(List<double> aInputs);
```

přepočítá bázi znalostí s aktuálními vstupními daty „List<double>“ (pole hodnot typu double, v programu reprezentováno jako pole uživatelských odpovědí, nebo vzor) a vrátí výstupní hodnoty „NetOutput“.

7.2.5 Tvorba vzorů

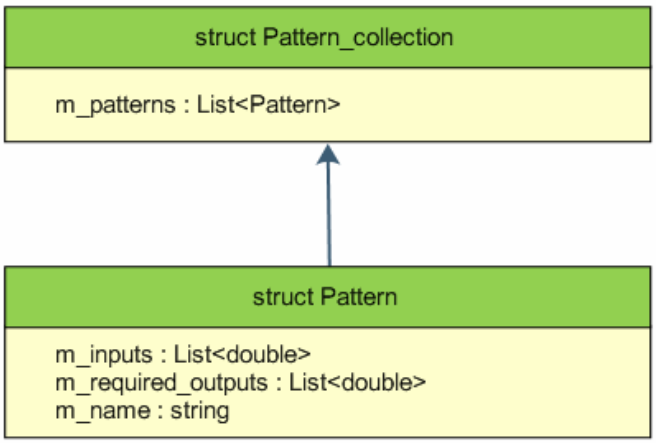
Vzory, ať už trénovací nebo testovací, reprezentují odezvu báze znalostí na vstupní informaci. Každý vzor je tedy tvořen množinou vstupních informací zde reprezentováno jako „List<double> m_inputs“ (viz. obrázek 35) a množinou výstupních reakcí báze znalostí „List<double> m_outputs“. Pro přehlednou správu vzorů je každému vzoru možné přiřadit název m_name. Jednotlivé vzory jsou ukládány v poli m_pattern a serializací ukládány na disk. Načtením z disku a deserializací je můžeme opět použít pro učení nebo ověřování báze znalostí. Vzory nelze zaměnovat mezi různými bázemi znalostí, neboť vstupní hodnoty jsou vázány na typ vstupních uzlů báze. Vzory jsou tvořeny v editoru modulu Patterns, který je zobrazen na obrázku 39.



Obrázek 39: Modul tvorby vzorů (trénovacích i testovacích)

7.2.5.1 Datové struktury

Každý vzor tvoří strukturu vstupních hodnot, výstupních hodnot a svého jména (*struct Pattern*). Tyto struktury jsou ukládány v poli *m_patterns*. Z důvodu serializace při ukládání vzorů je pole vzorů umístěno do struktury *Pattern_collection*.



Obrázek 40: Datové struktury vzorů

7.2.6 Modul učení

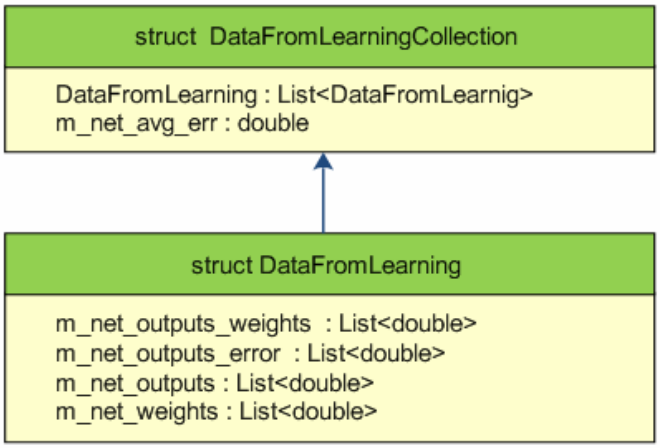
Učení báze znalostí probíhá podle algoritmu popsaného v kapitole 5.

7.2.6.1 Metody a funkce

Vstupními daty učicí funkce

```
public DataFromLearningCollection
LearnNet(Patern_colection aPaternCollection, double
aLearnRate);
```

jsou pole trénovacích vzorů, podle kterých se budou adaptovat váhy báze znalostí, a hodnota koeficientu učení (learning rate, v popisu učícího algoritmu označena jako μ), neboli rychlost konvergence. Výstupními daty funkce je struktura dat učení a střední absolutní chyba sítě. Výstupní strukturu funkce ukazuje obrázek 41.



Obrázek 41: Datové struktury učícího algoritmu

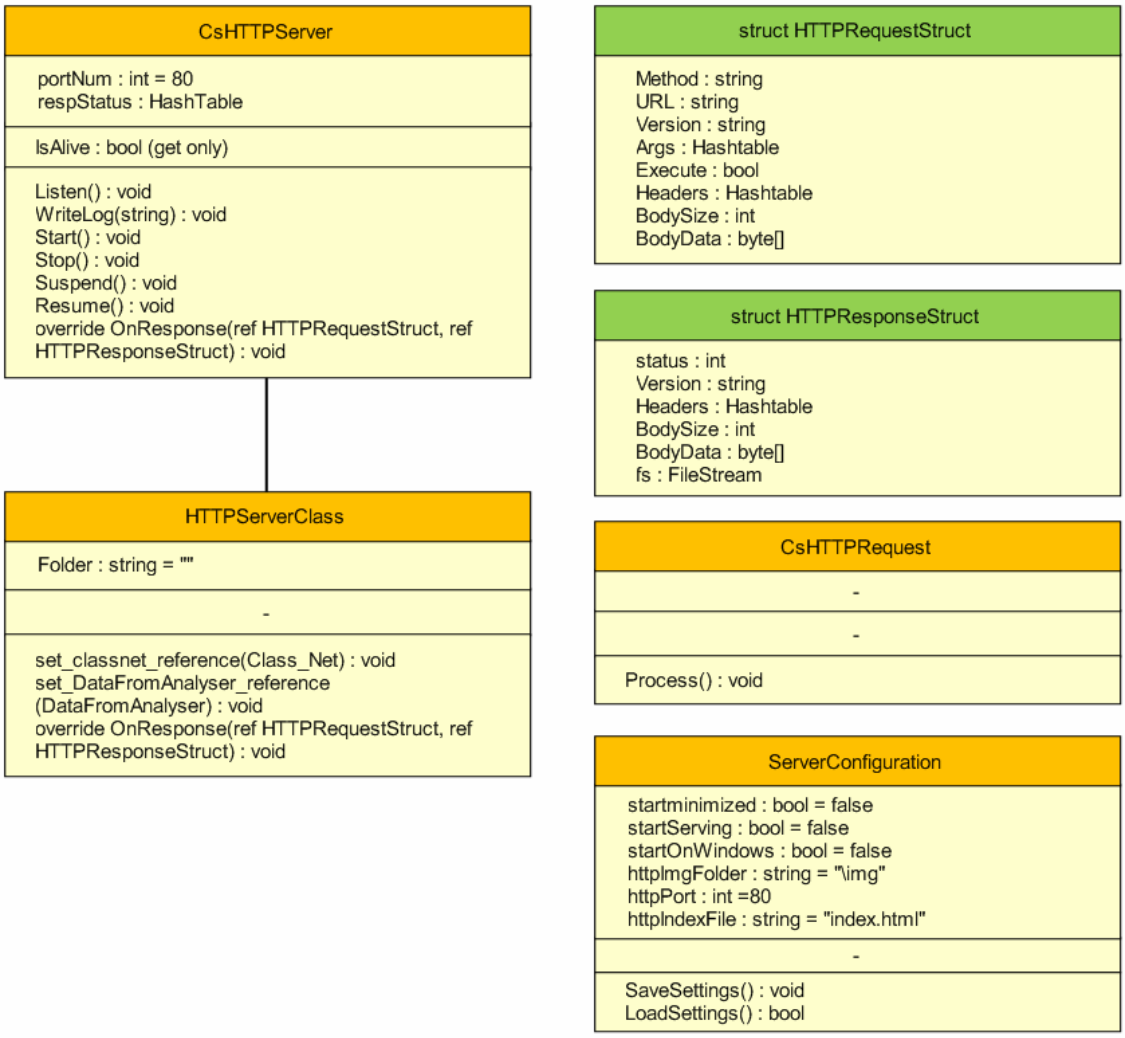
Funkce zajišťuje jednu epochu učení. Pro kompletní učicí proces je nutné ji zacyklit s vhodnou podmínkou ukončení cyklu (v programu dosažením chyby báze znalostí menší než stanovené, nebo dosažením stanoveného maximálního počtu epoch učení). Tento systém, ačkoliv není nejvhodnější z hlediska rychlosti, byl použit, protože je tak možné sledovat chování učícího procesu v jeho průběhu. Pro komerční využití by bylo vhodné tuto funkci optimalizovat. Například strukturu *DataFromLearnig*, která v průběhu učení ukazuje vývoj chyb a vah na vybraných uzlech a pravidlech báze znalostí, by bylo možné zcela odebrat. Struktura slouží výhradně k analýze chování učícího algoritmu a v průběhu učení generuje obrovské množství dat.

7.2.7 Komunikační rozhraní

Komunikační rozhraní je v programu realizováno dvěma způsoby. Expertní rozhraní je realizováno přímo v hlavní aplikaci jako GUI pomocí obrazovek jednotlivých modulů. Tyto rozhraní obsahují i modul pro konzultaci nad bází znalostí pro kontrolu správnosti konzultace. Konzultace v expertním režimu se spouští v modulu *Learning*.

Uživatelské rozhraní je realizováno pomocí vestavěného web serveru. Server se spouští z hlavního menu položkami *Consultation → Start*. Server pracuje v samostatném programovém vlákně a je tedy na hlavní aplikaci nezávislý. Je tedy možné pracovat s bází znalostí (i jinou než je využívána ke konzultaci) bez ovlivnění serveru. Spuštěním serveru se načte do paměti aktuální báze znalostí z hlavní aplikace. Položkou hlavního menu *Consultation → Stop* server zastaví činnost a je odstraněn z paměti včetně báze znalostí.

7.2.7.1 Datové struktury



Obrázek 42: Datové struktury serverového rozhraní expertního systému RESLA

7.2.7.2 Metody a funkce

Server je nakonfigurován parametry ze struktury *ServerConfiguration* a spuštěn v novém vlákně metodou

```
public void Listen();
```

třídy *CsHTTPServer*. Tato metoda poslouchá na portu *portNum* a v případě dotazu na server vytvoří nové vlákno ve kterém spustí metodu

```
public void Process();
```

třídy *CsHHTPRequest*. Metoda *Process*() převezme datový tok od klienta, zpracuje jeho požadavek do struktury *HTTPRequestStruct* a zavolá metodu *OnResponse*() třídy *HTTPServerClass*, které předá odkaz na strukturu *HTTPRequestStruct* a odkaz na strukturu odpovědi serveru *HTTPResponseStruct*. Funkce *OnResponse*() je výkonnou částí webového rozhraní expertního systému. Na základě zaslané odpovědi na daný dotaz expertního systému nebo příkaz klienta odeslaný tlačítky funkce sestaví vstupní data pro bázi znalostí. Vloží tyto data do báze znalostí a přepočítá ji metodami

popsanými v kapitole 7.2.4. Všechny metody potřebné k inferenci nad bází znalostí jsou součástí objektu báze znalostí (instance třídy *Class_Net*) předaného serveru. Z výsledků sestaví webovou stránku, kterou vloží s ostatními parametry do struktury *HTTPResponseStruct*. Z dat struktury *HTTPResponseStruct* je vytvořen stream (datový tok), který je odeslán nazpět klientovi. Nově vytvořené vlákno je uzavřeno a čeká se na další dotaz na server. V případě požadavku klienta na obrázek či jiný soubor, vrací funkce *OnResponse*() data tohoto souboru.

7.2.8 Nápověda

Nápověda k programu je realizována pomocí strukturovaných html stránek. Vyvoláním nápovědy z hlavního menu (Help → Index) otevře program hlavní stránku nápovědy v internetovém prohlížeči registrovaném v operačním systému. Zde je již nápověda v plné režii daného prohlížeče. Provázanost stránek je dána hypertextovými odkazy na příslušné oddíly a stránky. Z každé stránky je možné se dostat jednoduše zpět na index.

8 Závěr

Algoritmy přímého ladění vah pravidel pro báze znalostí expertních systémů, které vznikly v rámci této disertační práce, umožňují výrazně zrychlit a zjednodušit tvorbu bází znalostí expertních systémů a tím je lépe přizpůsobit potřebám praxe.

Tyto algoritmy ladění vah pravidel přímo adaptují váhy v pravidlové bázi znalostí a umožňují tak využít data pro ladění báze znalostí (učení na základě vzorů) při zachování všech výhod plynoucích z použití pravidel. Mezi hlavní výhody pravidel patří výborná srozumitelnost výsledné báze znalostí, modularita a jednoduchost.

Oproti metodám založeným na kombinaci neuronové sítě a pravidel jsou navržené metody přímého ladění vah v bázi znalostí jednodušší a rychlejší, neboť nevyžadují transformace mezi různými reprezentacemi znalostí.

Motivace disertační práce

Metody popsané v kapitole 2 reprezentují představu hybridních systémů tvorby a ladění bází znalostí, které kombinují pravidla a neuronové sítě. Využívají výhod obou typů reprezentací znalostí, ale jsou zatíženy velmi obtížným převodem těchto reprezentací znalostí navzájem.

Metody extrakce pravidel z neuronové sítě vytvářejí pravidla analýzou vah neuronů a struktury naučené neuronové sítě. Nebudeme-li se zabývat faktem, že tento proces je značně výpočetně náročný, jde o proces, při kterém je vytvářen model reálné domény neuronovou sítí. Následně je tato neuronová síť modelována pravidly, což snižuje přesnost výsledného pravidlového modelu vůči pozorované reálné doméně. Je totiž vytvářen model modelu. Mnohé z těchto algoritmů také neposkytují pravidla v příliš srozumitelné formě (např. M of N, popis aktivačními oblastmi neuronů, fuzzy množiny, atd.).

Ladění vah báze znalostí s diferenciálním evolučním algoritmem

V kapitole 4 je navržen algoritmus ladění vah pravidel v bázi znalostí diferenciálním evolučním algoritmem. Algoritmus ladění je navržen pro nejpoužívanější jednovrstvou strukturu báze znalostí expertního systému NPS32 popsanou v kapitole 4.1.

Bázi znalostí expertního systému NPS32 tvoří orientovaný graf pravidel s kontextovými vazbami. Tato pravidla využívají vlastní matematický aparát ke zpracování neurčitosti a šíření informace pomocí dvou hodnot (T, F). Jsou tak získány informace nejen o míře pravdivosti, či nepravdivosti hypotéz, ale i informace o nepřítomnosti informací nebo sporu v bázi znalostí.

V kapitole 4.2.2 je uveden důkaz, že kontextové vazby použité k řízení inference bází znalostí nemají vliv na odvozované výsledky. Zjednoduší se tak návrh učicího algoritmu, neboť není třeba zohledňovat kontextové vazby.

Algoritmus ladění vah pravidel nastavuje váhy v bázi pravidel pomocí diferenciálního evolučního algoritmu tak, aby pro daný vstupní vektor hodnot byla odchylka výsledné hodnoty cílového uzlu od požadované hodnoty co nejmenší.

Kvantifikace odchylek hodnot výstupů báze znalostí od požadovaných hodnot je realizována navrženou kriteriální funkcí. Kriteriální funkcí byla zvolena Euklidova vzdálenost neurčitosti cílových uzlů od požadovaných hodnot neurčitosti, neboť toto kritérium upřednostňuje odchylky s větším vlivem a urychluje tak konvergenci algoritmu k hledanému řešení.

Pro výpočet kriteriální funkce byla v kapitole 4.2.1 odvozena obecná funkce pro výpočet vlivu libovolného množství pravidel na cílový uzel. Odvozená funkce využívá speciální matematický aparát expertního systému NPS32.

Algoritmus ladění diferenciální evolucí byl implementován do stávajícího prázdného expertního systému NPS32. Byly vytvořeny a implementovány dva nové moduly tohoto expertního systému. První modul slouží k tvorbě a správě vzorů pro příslušnou bázi znalostí, druhý modul implementuje vlastní učicí algoritmus. Programové zpracování je podrobně popsáno v kapitole 7.

Experimentální ověření ladění vah báze znalostí s diferenciálním evolučním algoritmem

Algoritmus ladění byl testován na dvojrozměrné funkci AND, s úplným souborem vzorů. Při tomto testu algoritmus nastavoval váhy s chybou menší než 0,01 %. Druhým, reálným, testem byla diagnostika poruchy automobilu. Zde odchylka nebyla větší než 0,07 %, při přibližně 300 iteracích. Pro oba případy jsou výsledky stabilní.

V závěru čtvrté kapitoly je uveden formální důkaz nemožnosti použití matematického aparátu expertního systému NPS32 pro učení vícevrstvých bází znalostí gradientními metodami. Z tohoto důvodu byl pro analytické řešení automatického ladění vah pravidel pro vícevrstvé, libovolně strukturované pravidlové báze znalostí navržen zcela nový expertní systém s vlastní strukturou báze znalostí a zpracováním neurčitostí.

Metoda přímého ladění vah pravidel v bázi znalostí

Kapitola 5 popisuje vyvinutou analytickou metodu přímého učení báze znalostí. Tato metoda je hlavním výsledkem disertační práce. Algoritmus použitý v této metodě umožňuje ladění vah pravidel ve vícevrstvých, libovolně strukturovaných bázích znalostí.

Báze znalostí expertního systému využívajícího metodu přímého ladění vah pravidel je založena na původním modelu pravidla ohodnoceného neurčitostí vyjadřující míru informačního přínosu pravidla. Pravidla jsou v bázi znalostí navázána do libovolně hluboké stromové struktury. V bázi znalostí se mohou vyskytovat tři typy pravidel. Konjunktivní a disjunktivní pravidlo s libovolným počtem antecedentů a agregační funkce (pravidlo), která zajišťuje postupnou agregaci informace z příslušných pravidel v průběhu konzultace nad bází znalostí. Matematické modely pravidel jsou uvedeny v kapitole 5.1. Díky nahrazení nediferencovatelných funkcí minima a maxima v konjunktivním a disjunktivním pravidle funkcemi t-normy a s-normy (v daném pořadí) v pravděpodobnostním tvaru, mohly být uplatněny gradientní metody při adaptaci vah pravidel v bázi znalostí.

Algoritmus přímého ladění vah pravidel je založen na modifikaci algoritmu back-propagation, který je přepracován pro danou strukturu báze znalostí a použité modely pravidel. Jde o metodu učení s učitelem implementující delta pravidlo, což je pravidlo využívající k úpravě vah (resp. váhy) gradient chybové funkce. Vztahy pro adaptaci vah pravidel při ladění báze znalostí jsou odvozeny v kapitole 5.2.

Chybovou, kriteriální funkcí, byla zvolena Euklidova vzdálenost neurčitosti cílových uzlů od požadovaných hodnot neurčitosti, neboť toto kritérium upřednostňuje odchylky s větším vlivem a urychluje tak konvergenci algoritmu k hledanému řešení.

Vytvořený expertní systém RESLA

Pro možnost testování algoritmu přímého ladění vah pravidel v bázi znalostí byl vytvořen nový expertní systém RESLA do kterého byl algoritmus přímého ladění vah pravidel implementován.

Vytvořený expertní systém RESLA je složen z 10 modulů jejichž základ odpovídá standardní architektuře expertních systémů. Moduly programu tvoří tři logicky a programově ucelené části, a to moduly komunikace (HTTP server, Modul konzultace a Nápověda), jádro expertního systému (Báze znalostí, Interní analýza struktury báze, Inferenční mechanizmus, Modul učení) a grafické prostředí (Hlavní aplikace, Modul tvorby struktury báze, Modul editace struktury báze a oba moduly tvorby vzorů). Jednotlivé části programu jsou naprogramovány zcela odděleně s jasně definovanými rozhraními a mohou být zahrnuty do hlavního programu, nebo být součástí programu jako knihovna DLL (takto je v programu řešeno jádro expertního systému).

Mezi moduly, které jsou nad rámec standardní architektury expertních systémů patří modul HTTP server, který reprezentuje web server umožňující konzultace v expertním systému přes síť (LAN, nebo VAN). Dále interní analýza struktury báze, což je algoritmus analyzující strukturu a konzistentnost báze znalostí. Analyzátor samostatně vyhodnocuje, které uzly báze znalostí jsou vstupní, výstupní a skryté a stanoví optimální pořadí pro přepočet báze. Modul učení implementuje učicí algoritmus, modul tvorby vzorů vytváří vzory nastavením vstupních údajů báze znalostí a vyplněním požadovaných výstupů.

Experimentální ověření metody přímého ladění vah pravidel v bázi znalostí

Algoritmus přímého ladění vah pravidel v bázi znalostí byl testován na třech rozdílných úlohách.

Báze znalostí - logická úloha

První z úloh je báze znalostí realizující logické funkce AND, OR a XOR pro ověření funkčnosti a správnosti algoritmu. Průměrná chyba klasifikace báze znalostí pro logickou úlohu byla menší než 1 % pro trénovací vzory, pro nastavení koeficientu učení 0,1 i 0,4. Chyba klasifikace báze znalostí pro testovací vzory z tabulky 6 byla také menší než 1 %. Z grafů na obrázcích 18 až 21 je vidět vliv koeficientu učení na rychlost adaptace vah a rychlost poklesu globální chyby báze znalostí. Pro vyšší hodnoty koeficientu učení je adaptace vah pravidel rychlejší, což je způsobeno větším vlivem chybové hodnoty (rozdíl výstupní a požadované hodnoty báze) šířené zpětně bází pravidel. Pro komplexní báze znalostí byla vhodná hodnota koeficientu učení experimentálně stanovena na 0,1.

Báze znalostí pro diagnostiku poruch srdečního rytmu

Druhou úlohou je poměrně velká a komplexní báze znalostí z lékařského prostředí, na které bylo testováno chování algoritmu na složitých úlohách. Báze znalostí se věnuje problematice diagnostiky bradyarytmií srdečního rytmu na základě parametrů signálu EKG. Báze znalostí je tvořena 8 dotazovatelnými vstupními uzly a 15 cílovými uzly. Vlastní struktura báze znalostí je tvořena dalšími 60 uzly, které tvoří 32 pravidel

ve třech vrstvách. Báze znalostí byla naučena na 15 vzorů z tabulky 7 ve 42 učících cyklech s přesností lepší než 1 %. Naučená báze znalostí byla testována na 20 testovacích EKG záznamech. Průměrná chyba přes všechny testovací vzory a všechny výstupy byla 2 %. Největší odchylka činila pro jeden testovací vzor 12 %, nejmenší odchylka přes všechny vzory <1 %, medián <1 %.

Báze znalostí pro podporu při výběru odrůdy pšenice ozimé

Clem třetí úlohy bylo porovnat přístupy k tvorbě bází znalostí a schopnosti generalizace báze znalostí u profesionální báze znalostí vytvořené expertem a znalostním inženýrem a báze znalostí odladěné algoritmem přímého ladění vah pravidel. Báze znalostí představuje systém pro podporu při výběru vhodné odrůdy pšenice ozimé pro pěstování v různých lokalitách a různých klimatických podmínkách. Pro tuto úlohu byla vybrána báze znalostí Pšenice ozimá, vytvořená experty z oblasti zemědělství Ing. Zdeňkem Kryštofem a Ing. Janem Ksiažkiewiczem, CSc. v expertním systému NPS32 a odladěna ve spolupráci DITANA spol. s r.o., Velká Bystřice. Báze znalostí obsahuje 64 odrůd povolených v roce 2003 pro pšenici obecnou ozimou. Báze znalostí se skládá ze 14 dotazovatelných vstupních uzlů a deseti výstupních uzlů. Báze je tvořena dalšími 160 uzly, které tvoří 149 pravidel. Všechny výstupní uzly báze znalostí jsou realizovány agregačními funkcemi, které agregují postupně získávanou informaci z pravidel při zodpovídání otázek zastoupených dotazovatelnými uzly.

Báze znalostí byla naučena na 23 trénovacích vzorů v 93 učících cyklech s chybou klasifikace 9 %. Naučená báze znalostí byla testována na 10 testovacích vzorech. Průměrná chyba přes všechny testovací vzory a všechny výstupy byla 8 %.

Vyšší chyba klasifikace báze znalostí oproti předchozím dvěma úlohám je způsobena dvěma faktory. Prvním faktorem je proces získávání vzorů. Vzory byly vytvořeny konzultacemi jiného expertního systému, který uplatňuje zcela odlišné zpracování neurčitostí a práci s nimi. Snadno tak nastane případ, kdy stejné struktury pravidel poskytují odlišné výstupy pro malé změny vstupních hodnot pravidel. Je to způsobeno různými vnitřními funkcemi pravidel. Druhým faktorem je podobnost struktury obou srovnávaných bází znalostí. Hlubším studiem původní báze znalostí bylo zjištěno, že struktura báze znalostí Pšenice ozimá neodpovídá zcela přesně popisu této báze znalostí, tak jak byla uvedena expertem v tabulce 9, podle které byly vytvářena struktura pro bázi znalostí expertního systému RESLA.

V závěru třetí úlohy je ověřen vliv přetrénování báze znalostí a závislost chyby klasifikace trénovacích a testovacích vzorů na počtu trénovacích vzorů.

Shrnutí

Algoritmy vytvořené v disertační práci ukazují možnosti ladění vah pravidel přímo ve struktuře báze znalostí. Zachovávají se tak výhody ověřitelnosti konzistentnosti báze znalostí vůči reálné doméně, je zachována modularita a jednoduchost báze a je umožněno ladění vah pravidel z dat pomocí vzorů. Dojde tak k výraznému zrychlení tvorby báze znalostí a usnadnění práce znalostního inženýra.

8.1 Další směry výzkumu

Následující vývoj práce by mohl být orientován na prozkoumání informační schopnosti pravidlových bází znalostí s rozdílnou hloubkou pravidel a různým zpracováním neurčitosti. Algoritmy pro tvorbu rozhodovacích stromů (ID3, CART, C4.5, atd. [30]), které jsou po úpravě schopné vytvořit strukturu pravidel pro bázi znalostí z dat, běžně tvoří stromy o *n* rozhodovacích úrovních. Ze studia bází znalostí vytvořených expertem a znalostním inženýrem pro expertní systém NPS32, z nichž některé obsahují desítky uzlů (báze Pšenice [43] a Škoda [2]) vyplývá, že všechny tyto báze mají pouze vstupní a výstupní vrstvu a jednu vrstvu pravidel a přesto dostatečně přesně modelují danou reálnou doménu. Výstupní vrstvu v těchto bázích znalostí tvoří pravidlo, které upravuje hodnotu výstupu na základě změny některého z pravidel vstupujících do tohoto uzlu. Pokud by se podařilo dokázat, že se všechny smysluplné problémy dají modelovat tímto způsobem, bylo by podstatně jednodušší vytvořit algoritmus učení této báze znalostí s jedinou vrstvou pravidel. Pro učení takto strukturovaných bází lze použít algoritmus popsaný v kapitole 4.

Literatura

[1] ANDREWS, R., DIEDERICH, J., and TICKLE, A. B.: Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8(6), pp 373-389, December 1995

[2] BARTOŠÍK, M.: Expertní systém. Diplomová práce. FEKT VUT Brno, Brno, 2004

[3] BERENJI, H. R.: Refinement of approximate reasoning-based controllers by reinforcement learning, Proceeding of the Eighth International Machine Learning Workshop, Evanston, USA, 1991, pp 475-479

[4] BOSE, R., NAGARAJA G.: Performance Studies on KBANN. Procedings of the Fourth International Conference on Hybrid Intelligent, IEEE Computer Society, 2004, ISSN 0-7695-2291-2

[5] BUZING, P.: Hybrid systems: Two examples of the combination of rule-based systems and neural nets, 2001, Dostupné z WWW: <http://pds.twi.tudelft.nl/~buzing/Articles/hybrid.doc>

[6] CRAVEN, M. W. and SHAVLIK, J. W.: Using sampling and queries to extract rules from trained neural networks. In *Proc. of the 11th International Conference on Machine Learning*, San Francisco, USA, 1994

[7] ČÁSTEČKA, M., GRAJCAR, M., JIRSÍK, V.: Expertní systém NPS a jeho využití v zemědělství. Aplikace umělé inteligence AI'89, Praha, 1989, ISBN 80-02-991113-3

[8] FU, L. M.: Integration of neural heuristic into knowledge-based inference. Connection Science, Vol. 1, No. 3, 1989, ISSN 09540091

[9] FU, L. M.: Rule learning by searching on adapted nets. Procedings of the ninth national conference on artificial intelligence, Anaheim, 1991, ISBN 0262510596

[10] GEVA, S.; WONG, M. T.; ORLOVSKI, M.: Rule extraction from trained artificial neural network with functional dependency. First International Conference on Knowledge-Based Intelligent Electronic System, Adelaide, Australia, 1997, pp 559-564, ISBN: 0-7803-3755-7

[11] HAMPTON, J. R.: EKG stručně, jasně, přehledně. Grada Publishing a.s.,2007, ISBN 80-247-0960-0

[12] HAMPTON, J. R.: EKG v praxi. Grada Publishing a.s.,2007, ISBN 978-80-247-1448-6

[13] HORIKAVA, S., FURUHASHI, M., UCHIKAWA, Y.: On fuzzy modeling using algorithm. IEEE Transactions on neural networks, Vol 3, No. 5, pp 801-806

[14] Léčba poruch srdečního rytmu [online], IKEM, c2006 [cit. 2008-09-15]. Dostupný z WWW: <http://www.ikem.cz/www?docid=1004490>

[15] JACOBS, R. A., JORDAN, M. I., NOWLAN, S. J., and HINTON, G. E.: Adaptive mixture of local experts. Neural Computation, 3, pp 79–87, 1991

[16] JAN, J.: Číslicová filtrace, analýza a restaurace signálů. Vědecké monografie, VUTIUM, Brno, 2002, ISBN 80-214-1558-4

[17] JENKINS, D., GERRED, S.: *ECG library* [online], c1995-2009 [cit. 2008-11-01]. Dostupný z WWW: <http://www.ecglibrary.com/ecghome.html>

[18] JIRSÍK, V., VALENTA, J.: Expertní systém pro výběr automobilů, Automatizace 5/2006, ISSN 0005-125X

[19] JIRSIK V., VALENTA J.: Expertní systémy v praxi. AT&P Journal plus 7/2005, HMH s.r.o, 2005, ISSN 1336-5010

[20] JIRSIK V., VALENTA J.: Expertní systém pro výběr automobilů, Inteligentní systémy pro praxi, AD&M, 2006, ISBN 80-239-6535-2

[21] KAMRUZZAMAN S. M., ISLAM M.: Extracting of symbolic rules from artificial neural networks. Proceedings of world academy of science, engineering and technology, vol. 10, 2005, ISSN 1307-6884

[22] KASABOV, N.: Foundations of neural networks, fuzzy systems, and knowledge engineering. MIT Press, Londýn, 1996, ISBN 0-262-11212-4

[23] KHAN, G. M.: EKG a jeho hodnocení. Grada Publishing, a.s., 2005, ISBN 80-247-0910-4

[24] KRATOCHVÍL, Z.: Automatické vytváření pravidel pro znalostní báze metodami tzv. soft-computing. Disertační práce. FEKT VUT Brno, Brno, 2001

[25] LearnTheHeart.com [online], c1994-2009 [cit. 2008-11-01]. Dostupný z WWW: <http://www.learntheheart.com/>

[26] MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J.: Umělá inteligence (1). Akademie věd České republiky, Praha, 1993, ISBN 80-200-0496-3

[27] MAŘÍK, V., ŠTĚPÁNKOVÁ, O., LAŽANSKÝ, J.: Umělá inteligence (2). Akademie věd České republiky, Praha, 1993, ISBN 80-200-0504-8

[28] MASOUKA. R.. WATANABE. N., KAWAMURA, A., OWADA. Y. and ASAKAWA, K. Neurofuzzy systems - fuzzy inference using a structured neural network, Proc. International Conference on Fuzzy Logic and Neural Networks, Lizuka, 1990, pp 173-177

[29] McMILLAN, C., MOZER, M.C., SMOLENSKY, P.: The connectionist scientist game: rule extraction and refinement in a neural network. Proceedinds of Thirteenth Annual Conference of the Cognitive Science Society, Hillsdale, NJ, USA, 1991

[30] MRLINA, Z.: Rozhodovací stromy. Diplomová práce, FEKT VUT Brno, Brno, 2006

[31] SAITO, K., NAKANO, R.: Medical diagnostic expert system based no PDP model. Procedings of IEEE international conference on neural network, San Diego, 1988

[32] SHAVLIK, J.: Combining symbolic and neural learning. Machine learning, vo. 14, Kluwer Academic Publishers, Boston, 1994, ISSN 0885-6125

[33] SETIONO R., BAESENS B., MUES C.: Recursive neural network rule extraction for data with mixed attributes. Neural Networks, IEEE Transactions on, 19(2), pp 299–307, 2008

[34] SETIONO R.: A penalty-function approach for pruning feedforward neural networks. Neural computation, vol. 9, no. 1, 1997, pp. 185-204

[35] STORN, R., PRICE, K.: Differential Evolution for Continuous Function Optimization, Dostupné z WWW: <http://www.icsi.berkeley.edu/~storn/code.html>

[36] THRUN S. B.: Extracting Provably Correct Rules from Artificial Neural Networks, University of Bonn, 1993

[37] TICKLE, A. B., ANDREWS, R., GOLEA, M. , and DIEDERICH, J.: The truth will come to light: directions and challenges in extracting the knowledge embedded within trained artificial neural networks. *Neural Networks, IEEE Transactions on*, 9(6), pp 1057–1068, 1998.

[38] TOWELL, G.: Symbolic knowledge and neural networks: Insertion, refinement, and extraction. Ph.D. thesis, Department of Computer Sciences, University of Wisconsin, Madison, 1991, Dostupné z WWW: <http://www.cs.wisc.edu/~shavlik/abstracts/towell.thesis.abstract.html>

[39] TOWELL, G., SHAVLIK, J.: Extracting refined rules from knowledge-based neural networks. Machine Learning, vo. 13, Kluwer Academic Publishers, Boston, 1993, ISSN 0885-6125

[40] TOWELL, G., SHAVLIK, J.: Knowledge-based artificial neural networks. Artificial intelligence, Vo. 70, Elsevier Science Publishers Ltd., Essex, 1994, ISSN:0004-3702

[41] VALENTA J.: Rule-based expert systems learning method. EEICT 2006, konference, VUT FEKT a FIT, 2006, ISBN 80-214-3163-6

[42] VALENTA, J. Modified back-propagation for rule based systems. In Proceedings of the IASK International Conference E-Activity and Leading Technologies. Porto, Portugal. 2007. pp 150-154, ISBN 978-972-99397-5-4

[43] VALÁŠEK, R.: Expertní systém NPS. Diplomová práce, KAMT FE VUT, Brno, 1990

[44] VOJTA, P.: Expertní systém NPS32. Diplomová práce, FEI VUT Brno, 2001

[45] VOLNÁ, E.: Neuronové sítě 2. Ostravská univerzita, Přírodovědecká fakulta, Ostrava, 2002, Dostupné z WWW: <http://ki.fpv.ukf.sk/mat**e**rialy_public/Um**e**la%20int**e**lig**e**ncia/n**e**uronov**e**sit**e**2.pdf>

[46] VALENTA, J., PANÁČEK, T.: Knowledge base learning for rule-based expert systems. In proceedings of Mendel 2006 - International Conference on Soft Computing. Brno, 2006, ISBN 80-214-3195-4

9 Publikované práce

9.1 Publikované práce vztahující se k tématu disertační práce

VALENTA, J. Expertní systém pro diagnostiku bradyarytmií srdečního rytmu. *Automatizace.* 2009. 52(10). ISSN 0005-125X (schváleno k otisknutí)

VALENTA, J. Learning algorithm for rule-based knowledge-bases. Brno, NOVPRESS s r.o. 2009. p. 52 - 56. ISBN 978-80-214-3870-5.

VALENTA, J. Modified back-propagation for rule based systems. In *Proceedings of the IASK International Conference E-Activity and Leading Technologies.* Porto, Portugal. 2007. p. 150 - 154. ISBN 978-972-99397-5-4.

VALENTA, J.; PANÁČEK, T. Knowledge base learning for rule-based expert systems. In *Mendel 2006.* Brno. 2006. p. 91 - 94. ISBN 80-214-3195-4.

VALENTA, J. RULE-BASED EXPERT SYSTEMS LEARNING METHOD. In *Proceedings of the 12th conference Student EEICT 2006 Volume 4.* Ing. Zdeněk Novotny, CSc., Ondráčkova 105, Brno. 2006. p. 486 - 490. ISBN 80-214-3163-6.

JIRSÍK, V., VALENTA, J. Expertní systém pro výběr automobilů. *Automatizace.* 2006. 49(5). p. 319 - 639. ISSN 0005-125X.

JIRSÍK, V.; VALENTA, J. Expertní systém pro výběr automobilů. In *Inteligentní systémy pro praxi.* Ostrava, AD&M, Urxova 470, Ostrava. 2006. p. 61 - 122. ISBN 80-239-6535-2.

JIRSÍK, V., VALENTA, J. Expertní systémy v praxi. *AT&P journal PLUS 5 2004.* 2005. 5(7). p. 5 - 12. ISSN 1336-5010.

9.2 Ostatní publikované práce

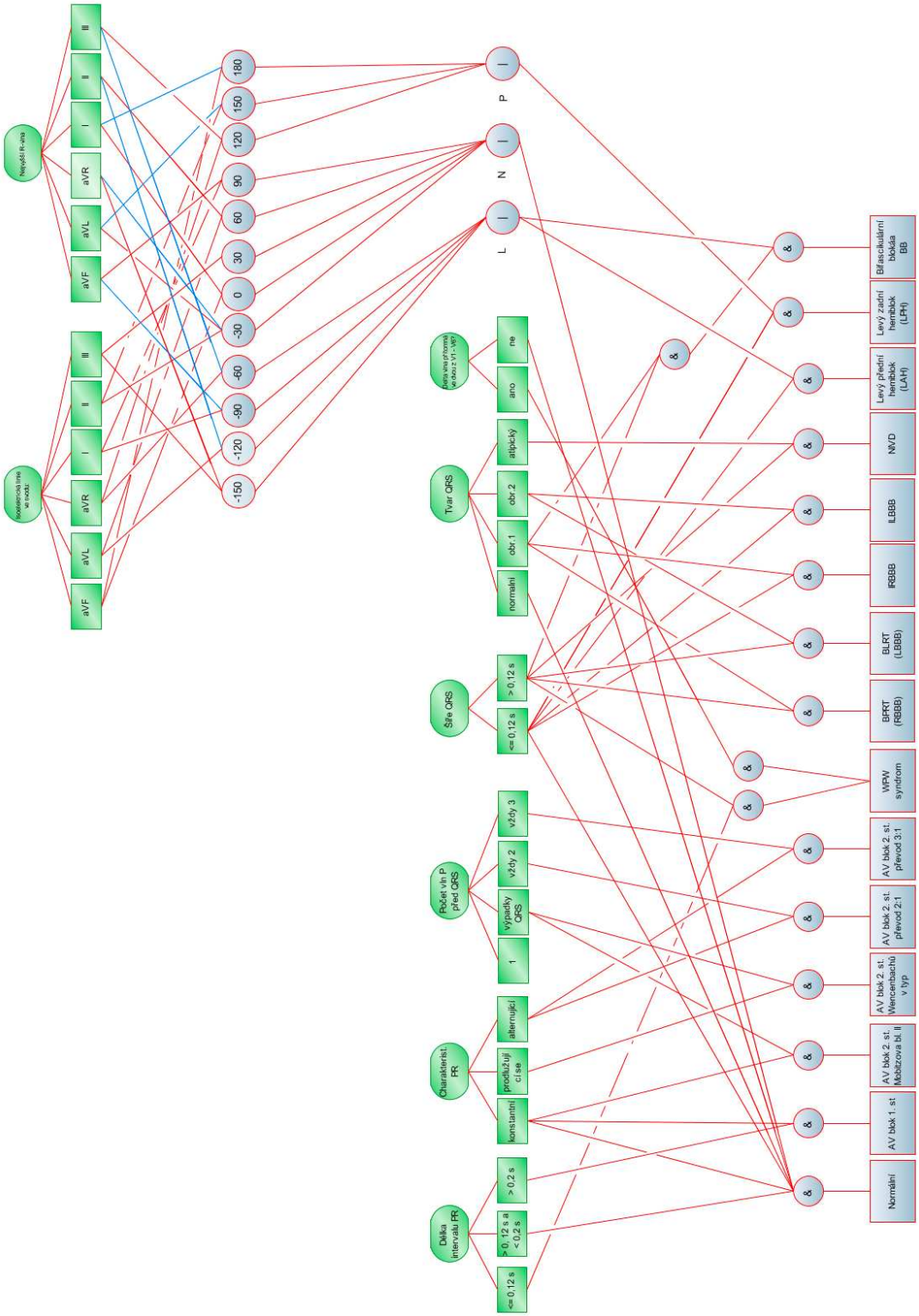
VALENTA, J.; PANÁČEK, T.; JIRSÍK, V. E-learning tools for artificial intelligence. In *Proceedings of the IASK International Conference E-Activity and Leading Technologies.* Porto, Portugal. 2007. p. 104 - 108. ISBN 978-972-99397-5-4.

SÁBLÍK, V.; VALENTA, J. TOOLBOX NEURONOVÝCH SÍTÍ PRO E-LEARNING. In *Sborník konference eLearning Hradec 2007.* Hradec Králové, Gaudeamus. 2007. p. 411 - 415. ISBN 978-80-7041-573-3.

PANÁČEK, T.; VALENTA, J. ARTIFICIAL INTELLIGENCE TOOLS FOR E-LEARNING. In *Proceedings of 13th International Conference on Soft Computing (MENDEL 2007).* 2007. p. 147 - 150. ISBN 978-80-214-3473-8.

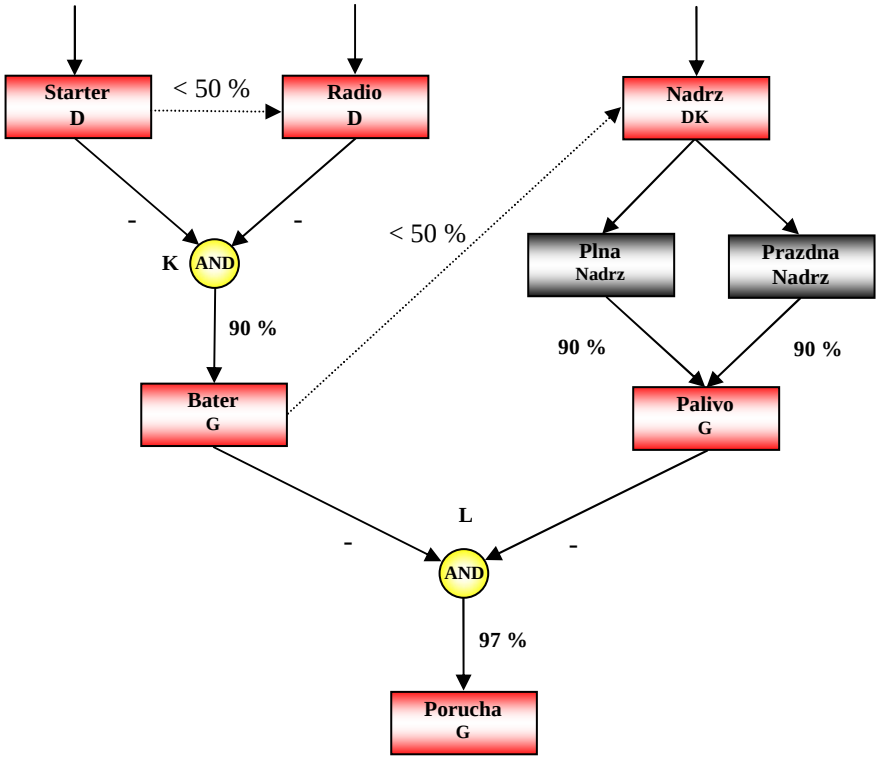
VALENTA, J. Knowledge-Base UAMT. In *Proceedings of the 13th conference Student EEICT 2007 Volume 4*. Ing. Zdeněk Novotny, CSc., Ondráčkova 105, Brno. 2007. p. 424 - 428. ISBN 978-80-214-3410-3.

10 Příloha A – Grafická reprezentace báze znalostí pro diagnostiku poruch srdečního rytmu



Grafická reprezentace modelové báze znalostí pro diagnostiku poruchy automobilu

Uvedená báze znalostí představuje strukturu modelové báze znalostí pro diagnostiku poruchy automobilu vytvořená v expertním systému NPS32. Báze znalostí je na obrázku 43.



Obrázek 43: Báze znalostí pro diagnostiku poruchy automobilu

Báze znalostí je tvořena dvěmi pravidly (K a L), které váží dva dotazovatelné přímé uzly (D) a jeden dotazovatelný kvantitativní (DK, reprezentuje výběr z odpovědí) na tři cílové uzly (G). V bázi znalostí jsou dvě kontextové vazby reprezentované tečkovanou šipkou. Tyto kontextové vazby slouží k řízení inference bází znalostí. Např. otázka zastoupená dotazovatelným uzlem „Radio“ se bude při konzultaci pokládat pouze v případě, že odpověď na otázku uzlu „Starter“ bude menší než 50 % podle tabulky 14 z kapitoly 7.2.2.

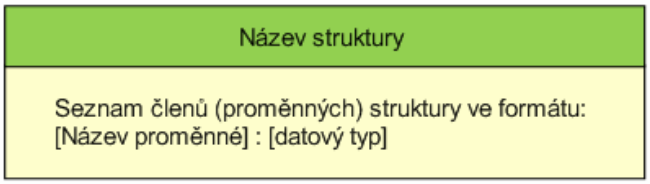
11 Příloha C – UML diagramy

V předložené disertační práci jsou popsány datové objekty obou programových implementací algoritmů ladění vah pravidel pomocí UML diagramů. V této kapitole je stručně popsáno jak číst UML diagramy obsažené v této disertační práci.

UML je univerzální grafický jazyk pro popis programových struktur, datových objektů a činnosti programů.

V předložené disertační práci se vyskytují dva typy datových objektů. Třída, jejíž popis je na obrázku 44 a struktura, jež je popsána na obrázku 45.

Obrázek 44: Diagram programové třídy (class)



Obrázek 45: Diagram programové struktury (struct)

Spojnice, které se vyskytují mezi třídami reprezentují hierarchii tříd, kde na nejvyšší úrovni je mateřská třída, ze které jsou děděny další třídy.